Ce User Guide

For ARPUS/Ce Version 2.5.1 (August 1999)

ARPUS/Ce - High Performance, High Productivity, Text Editing, and Terminal Emulation

Enabling Technologies Group, Inc.
8601 Dunwoody Place, Suite 300
Atlanta, Ga 30350
Support: (770) 64–ARPUS (770) 642–7787
Fax: (770) 993–4667
arpus@etginc.com

BEFORE YOU OPEN THE PACKAGE CONTAINING THE MEDIA, YOU SHOULD CAREFULLY READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. BY OPENING YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PART TO THIS LICENSE AGREEMENT. IF YOU FIND ANY OF THE TERMS OF THIS LICENSE AGREEMENT OBJECTIONABLE, PLEASE CONTACT ETG TO NEGOTIATE THEM. IF YOU AND ETG ARE UNABLE TO NEGOTIATE A MUTUALLY SATISFACTORY RESOLUTION OF YOUR OBJECTIONS, YOUR PLACE OF PURCHASE WILL REFUND YOUR LICENSE PRICE IF YOU RETURN THE UNOPENED PACKAGE CONTAINING THE MEDIA AS WELL AS YOUR SALES RECEIPT AND THE END USER DOCUMENTATION WITHIN THIRTY (30) DAYS OF PURCHASE.

LICENSE AGREEMENT

Subject to the provisions contained herein, Enabling Technologies Group, Inc. (”ETG”) hereby grants to you a non–exclusive license to use its proprietary computer software product (”Software”) contained in the package. As described below, your use of the Software, when enabled, is limited to a ”Number” of simultaneous users.

SOFTWARE AND DOCUMENTATION. ETG shall furnish the Software to you on media in machine–readable object code form and may also provide ETG’s standard documentation (”Documentation”) containing instructions for operation and use of the Software.

PASSWORD/REGISTRATION NUMBER. A password(s) or registration number, which enables the Software, may be included. If a password(s) or registration number is not included, and is applicable for the Software, ETG or its designee shall provide you with a password(s) or registration number corresponding to the number (”Number”) and type of licenses listed on your purchase order or ETG’s sales order or invoice.

LICENSE TERM AND CHARGES. The term of this license commences upon delivery of the Software to you and is perpetual unless earlier terminated upon default by one of the parties or as otherwise set forth herein. The license fee corresponds to the Number of concurrent users. If the ”Save” feature and/or a Number greater than one (1) applies to the Software and you desire to enable the ”Save” feature or to increase the size of the Number, then you may do so by notifying your place of purchase of your desire to enable the Software, or the desired Number or Number increase, and paying to your place of purchase the applicable price. Your place of purchase will then provide you with a password(s) or registration number which will enable the ”Save” feature or so effect the Number. **Notwithstanding the foregoing, if the Software and/or Documentation is being provided for demonstration or evaluation purposes only, then your license to use the Software shall automatically terminate thirty (30) days (or such longer period as authorized by ETG) after you commence using the Software, and you must return the Software and Documentation and all copies thereof.**

TITLE. Title, and ownership right, and intellectual property rights in and to the Software and Documentation shall remain in ETG and/or in suppliers to ETG of programs contained in the Software. The Software is provided for your own internal use under this license. This license does not include the right to sub license and is personal to you and therefore may not be assigned (by operation of law or otherwise) or transferred without the prior written consent of ETG. You acknowledge that the Software in source code form remains a confidential trade secret of ETG and/or its suppliers and therefore you agree not to attempt to decipher or decompile the Software or develop source code for the Software or knowingly allow others to do so. You also agree not to develop passwords or registration numbers, to alter the Number, or to determine how ETG develops such password(s) or registration number(s) or to otherwise enable the Software on equipment or for concurrent users not licensed or to knowingly allow others to do so.

UPDATES. This license does not grant you any right, license, or interest in and to any improvements, modifications, enhancements, or updates to the Software and Documentation. Updates, if available, may be obtained by you at ETG’s then current standard pricing, terms, and conditions. If the media inside the package is an update, then the Documentation provided may be limited in scope and a new password(s) or registration number may or may not be required or supplied by ETG.

LIMITED WARRANTY. ETG warrants that for a period of ninety (90) days from the date you purchased a license to it, the Software, if operated as directed, will substantially achieve the functionality described in the specifications. ETG does not warrant, however, that your use of the Software will be uninterrupted or that the operation of the Software will be error–free. ETG also warrants that the media containing the Software, if provided by ETG, is free from defects in material and workmanship and will so remain for ninety (90) days from the date you purchased a license to it. ETG’s sole and exclusive obligation and liability for any breach of this warranty shall be, at ETG’s sole discretion: (i) to repair or correct your defective media so that it will conform to the above warranty; or (ii) to replace your defective media: or(iii) to advise you how to achieve substantially the same functionality with the Software as described in the Documentation through a procedure different from that set forth in the Documentation; or (iv) to refund the purchase price paid by you for the license to the defective Software. Repaired, corrected, or replaced Software and Documentation shall be covered by this limited warranty for the period remaining under the warranty that covered the original Software, or if longer, for thirty (30) days after the date (a) of shipment to you of the repaired or replaced Software, or (b) ETG advised you how to operate the Software so as to avoid the difficulty that you were having in achieving functionality. Only if you inform ETG of your problem with the Software during the applicable warranty period will ETG be obligated to honor this warranty. You must contact ETG by telephone or by letter to inform ETG of the problem. THIS IS A LIMITED WARRANTY AND IT IS THE ONLY WARRANTY MADE BY ETG. ETG MAKES NO OTHER EXPRESS OR IMPLIED WARRANTY AND EXPRESSLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NO ETG DEALER, AGENT, OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATIONS, EXTENSIONS OR ADDITIONS TO THIS WARRANTY. If any modifications are made to the Software by you during the warranty period; if the media is subjected to accident, abuse, or improper use; or if you violate the terms of this Agreement, then this warranty shall immediately be terminated. This warranty shall not apply if the Software is used on or in conjunction with hardware or programs other than the unmodified version of hardware and programs with which the Software was designed to be used as described in the Documentation

SOME STATES OR JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY BY STATE OR JURISDICTION.

RESTRICTED USE. If the Software is provided for demonstration and evaluation purposes, it may be used by you for no longer than thirty (30) days (or such longer period authorized by ETG in writing), in order for you to determine whether to purchase a license to enabled Software. The Software with all features enabled and Documentation shall be used exclusively by you and your qualified employees and consultants. An employee or consultant is qualified only if he or she has executed an agreement or else is otherwise bound to abide by the restrictions imposed upon you herein under "Title" and "Restricted Use." Unless the Number is greater than one (1), you may use the Software on only one (1) single computer at any time (a "computer" hereby defined as one (1) central processing unit and associated peripheral equipment, all at one (1) location). No more than the Number may use the Software concurrently. ETG hereby consents to your making a reasonable number of backup or archival copies of the Software. All such copies shall be subject to this Agreement and shall contain all of ETG's notices regarding copyrights, trademarks, other proprietary rights, and restrictions on users of disabled copies that are contained in the Software ETG originally provided to you. You may not modify, disassemble or reverse engineer the Software or make derivative or compilation works of the Software.

SUPPORT AND TRAINING. Technical support and training for the Software, if available, shall be provided at ETG's then current standard pricing, terms, and conditions.

LIMITATION OF LIABILITY. Your sole and exclusive remedies for any damage or loss in any way connected with the Software, whether by ETG's breach of any express or implied warranty, negligence, or any breach of any other duty, shall be, at ETG's option, repair or replacement of the Software or return for credit or refund of the net book value of a license to such Software or documentation. Under no circumstances and under no legal theory, tort, contract or otherwise, shall ETG be liable to you or any other person for any indirect, special, incidental, or consequential damages of any character including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, or for any damages in excess of ETG's list price for a license to the Software and Documentation, even if ETG shall have been informed of the possibility of such damages, or for any claim by any other party. ETG's third party program suppliers make no warranty, and have no liability whatsoever to you.

SOME STATES AND JURISDICTIONS DO NOT ALLOW THE LIMITATIONS OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

TERMINATION. Either party may terminate this Agreement immediately in the event of default by the other party. Upon any termination of this Agreement, you shall immediately discontinue the use of the Software and shall within ten (10) days return to ETG all copies of the Software and Documentation. You may also terminate this Agreement at any time by destroying the Software and Documentation and all copies thereof. Your obligations to pay accrued charges and fees shall survive any termination of this Agreement.

MISCELLANEOUS. This Agreement represents the complete and exclusive statement of the agreements concerning this license between the parties and supersedes all prior agreements and representations between them. It may be amended only by a writing executed by both parties. THE ACCEPTANCE OF ANY PURCHASE ORDER PLACED BY YOU FOR THE SOFTWARE AND/OR DOCUMENTATION IS EXPRESSLY MADE CONDITIONAL ON YOUR ASSENT TO THE TERMS AND CONDITIONS SET FORTH HEREIN, AND ETG AGREES TO FURNISH THE SOFTWARE AND DOCUMENTATION ONLY UPON THESE TERMS AND CONDITIONS AND NOT THOSE CONTAINED IN YOUR PURCHASE ORDER. If any provision of this Agreement is held to be ineffective, unenforceable, or illegal under certain circumstances for any reason, such decision shall not affect the validity or enforceability (i) of such provision under other circumstances or (ii) of the remaining provisions hereof under all circumstances and such provision shall be reformed to and only to the extent necessary to make it effective, enforceable, and legal under such circumstances. All headings are solely for convenience and shall not be considered in interpreting this Agreement. This Agreement shall be governed by and construed under Georgia law except as required by U.S. Government rules and regulations to be governed by Federal law.

U.S. Government Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (a) through (d) of the Commercial Computer–Restricted Rights clause at FAR 52.227–19 when applicable, or in subparagraph (c) (1) (ii) of the Rights in Technical Data and computer Software clause at DFARS 252.227–7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Enabling Technologies Group, Inc., P.O. Box 500038, Atlanta, GA 31150.

Ce User Guide

CONTENTS **PAGE**

ARPUS/*Ce*: Introduction

Getting Out

If you have started ***Ce*** without running “ce_init” and are looking at the manual to determine how to get out without modifying the file, **read this**: Under normal conditions “<Ctrl> n” or “<Ctrl> F8” will exit without saving the file. If this has no effect, use the mouse to move the cursor to the “command:” window and type “wc –q” (window close –quit). You may get the prompt “File modified, OK to quit?”. Respond with a “Y”. If you have to use the “wc –q” command, you will need to execute “ce_init” and then “ce –reload”.

ARPUS/Ce is an integrated ASCII text editor and X based terminal emulator modeled after the editor and terminal emulator on Apollo Domain systems. The ce tool is a robust and extensible programmer’s editor. The ceterm tool is meant to be a replacement for vendor–supplied X–based terminal window programs such as IBM’s “aixterm”, HP’s “hpterm”, DEC’s “decterm”, and Sun’s “Shell Tool”.

If you are a new ***Ce*** user the very first thing you need to do is run the shell script “ce_init” to build your $HOME/.Cekeys file and set some options in $HOME/.Xdefaults. The “ce_init“ command provides a short getting started tutorial and references several start up help pages. If you are just installing ***Ce***, look at the README file in the ***Ce*** directory created by un–tar’ing the ***Ce*** install file or see *Appendix E, Installation Summary*. If you are familiar with ASCII text editors such as vi and dtpad, see the *Ce Quick Start Reference* at the end of this section to try out the basic functionality of ***Ce***.

If you are a new user of ARPUS/Ce, we encourage you to learn first how to use this product as an editor. *Chapters 1 through 3* are devoted entirely to editor functions. In *Chapter 4* we introduce you to ceterm. This progression allows you to smoothly transition from your current shell tool to ceterm.

The bulk of the documentation you will use is located in the help files. The help files go into much greater detail than this manual. Pressing the “F9” key will prompt for a help topic. On Sun workstations, the “Help” key also works. A null response will bring up an introduction to using help. Topic “commands” will bring up an index of all available topics.

Conventions

Ce	Refers to the product ***Ce*** as a whole.
ce, ceterm, cv	Refers to the UNIX commands used to invoke the product.
“Command:” window	Refers to the input window in the lower left corner of a ***Ce*** window which has the prompt “Command:”, and is used to enter ***Ce*** commands.
“Command:” prompt	Examples using the “Command:” prompt as a prefix are executed in the command: window.
“UNIX:” prompt	Examples using the “UNIX:” prompt are UNIX shell commands.

User Guide Organization

Chapter 1 — **Editing Basics**: Shows how to open a file for editing and introduces ***Ce***'s basic editing features (cursor control, window scrolling, pattern search, text editing, search–and–replace, and copy/cut/paste operations, window geometry, on–line help, bug reports and enhancement requests).

Chapter 2 — **Environment Customization**: Shows how to customize certain aspects of the ***Ce*** environment, such as foreground/background colors, window size and location, tab processing, and other edit environment characteristics.

Chapter 3 — **Advanced Editor Use**: Gives in–depth descriptions of ***Ce*** editing concepts such as regular pattern usage, copy/cut/paste operations, edit command scripts, tab processing, and file saving.

Chapter 4 — **The "ceterm" Window**: Describes the "ceterm" terminal emulator.

Appendix A — **Key summary**: Lists all keys that have default ***Ce*** definitions assigned.

Appendix B — **Command Summary**: Lists all ***Ce*** commands along with brief descriptions of their functions.

Appendix C — **Key Customization**: Discusses the default key definitions and shows how to modify those definitions or create new definitions.

Appendix D — **Change History:** Summarizes the changes between releases of ***Ce***.

Appendix E — **Installation Summary**: Lists the files, directories, and links, both in the system directories and in the users' home directories, that are used by ***Ce***.

Appendix F — **License Server:** Describes the setup for the license server. This is of interest to the person installing the product.

We encourage you to seat yourself at a workstation and use ***Ce*** while you browse this manual. You will find numerous command examples scattered throughout the User Guide, since it is most often the case that hands–on experimentation will greatly increase the speed with which you gain proficiency with ***Ce***.

You will also find that the on–line files include extensive help text for all ***Ce*** commands and concepts. As indicated in *Chapter 1, Editing Basic*s, such help is generally available at the click of a mouse button.

You have to run "ce_init" before running ce or ceterm the very first time.

Ce Quick Start Reference

Command	Definition
F1	Mark (See Note)
<Shift> F1	Mark Rectangular
F2	Copy to Paste Buffer
<Shift> F2	Copy to Paste Buffer Rectangular
F3	Cut to Paste Buffer
<Shift> F3	Cut to Paste Buffer Rectangular
F4	Paste
<Shift> F4	Paste Rectangular
F5	Delete Line
<Shift> F5	Delete to End of Line
<Ctrl> F5	Delete Word
F6	Paste Line Deleted by F5
<Shift> F6	Paste Area Deleted by <Shift> F5
<Ctrl> F6	Paste Word Deleted by <Ctrl> F5
F7	Undo (Infinite undo stack)
<Shift> F7	Redo (Reverse of undo)
F8	Save and Quit (window manager Close pulldown does this)
<Shift> F8	Save file, don't quit. Toggle to read only mode
<Ctrl> F8	Quit without saving
F9	Help
<Ctrl> t	Move to top of file
<Ctrl> b	Move to bottom of file
<Ctrl> m	Toggle read only mode
Insert	Toggle Insert/overstrike mode
Del	Delete a character
Home	Go to beginning of line
End	Go to end of line

Command	Definition
<shift> Left	Scroll Left one character
<shift> Right	Scroll Right one character
<Ctrl> left	Scroll Left ten characters
<Ctrl> right	Scroll Right ten characters
Arrow keys	Move the cursor as expected
Page up/down	Page by 1/2 page

NOTE: The mark command “F1” is like pressing the left mouse button down in an xterm. It establishes the base for the region to be copied or cut and begins text highlighting. The copy command “F2” is like releasing the left mouse button in an xterm. Note that a rectangular mark implies a rectangular copy or cut.

SUN Type 5 Keyboard Extensions

Command	Definition
Help	Help
Again	Copy from text cursor to end of line to command window (Ceterm –> UNIX command, Ce –> ***Ce*** command)
Props	Show keyname for next key pressed (kk command)
<Shift> Props	Show keyname for next 2 keys pressed (kk;kk)
Undo	Undo
<Shift> Undo	Redo
Front	(Intercepted and processed by SUN window manager)
Open	(Intercepted and processed by SUN window manager)
Copy	Copy to Paste Buffer
<Shift> Copy	Copy to Paste Buffer Rectangular
Paste	Paste
<Shift> Paste	Paste Rectangular
Cut	Cut to Paste Buffer
<Shift> Cut	Cut to Paste Buffer Rectangular

NOTE: “<Ctrl> Copy”, “<Ctrl> Cut”, and “<Ctrl> Paste” perform the expected operations to an alternate paste buffer. This makes it easy to grab two separate chunks of text. “<Ctrl> F2”, “<Ctrl> F3”, and “<Ctrl> F4” also access the alternate paste buffer.

Chapter 1: Editing Basics

Organization of This Chapter

This chapter will take you through a simple ***Ce*** edit session to introduce you to basic ***Ce*** operations. We strongly encourage you to follow the examples by using ***Ce*** while you read this chapter.

This chapter describes the following:

- ***Ce*** shell commands and your search path
- Initialization of your personal ***Ce*** environment
- Creation of a ***Ce*** session using a practice file
- The appearance of a ***Ce*** session on your display
- Cursor control and window scrolling
- Basic text editing operations
- Cut/copy/paste operations
- Search and search/replace operations
- Window geometry
- On–line help
- Bug reports and enhancement requests

These sections are intended to be introductory in nature. Subsequent chapters present information on commands and concepts in greater detail.

Ce Shell Commands and Your Search Path

Type the following command in your shell.

Command: % ce –version

If ***Ce*** is already in your search path, it will execute output that looks something like this:

ARPUS/ce, Version 2.5 Compiled: Nov 26, 1997 11:07:04

If ***Ce*** is not in your path you will receive a message like this:

ce: Command not found.

In this case, you will need to ask your system administrator where the ***Ce*** commands are located, and add to your shell's “$PATH” environment variable the directory that contains those commands.

If you execute ***Ce*** and an empty window comes up for a while and then goes away with a message about not being able to get a license, either the license server is not running or you do not have your LSHOST environment variable set to the node running the license server. See *Chapters 9 and 10* for more information.

Initialization of Your Personal Ce Environment

Ce allows you to assign edit commands and command combinations to specific keys. For your convenience, ***Ce*** includes a set of key definitions already developed for your keyboard. It is possible that your system administrator or someone else has set up a ".Cekeys" file in your home directory already. Check to see if that file exists. If so, you can probably skip this step. Find out who created the file for you and what changes they made from the default definitions. Before using ***Ce***, you should run the "ce_init" program to copy the default key definitions into your home directory using the following command:

UNIX: ce_init

"ce_init" will create a file in your home directory called ".Cekeys". Chapter 8, Key Customization, describes the use of that file. For the remainder of this chapter, you do not need to concern yourself with the contents of that file, only that it is properly installed in your home directory. If you have run ce before you ran "ce_init", you will need to run "ce –reload" one time.

If you use more than one type of machine (for example Sun and HP) and you use the same home directory for both machines, you will need to have a separate ".Cekeys" file for each platform. This only applies if you physically sit in front of each type of machine. If you always sit at the same workstation and just telnet or rlogin to other machines you only need key definitions for your machine. The reason for multiple sets of key definitions on multiple machine types is that the keyboards are different. The environment variable CEKEYS allow you to change the name of the ".Cekeys" file. Run "ce_init" on one platform and then rename the created ".Cekeys" file to some useful name such as ".Cekeys.hp". In your .profile, ".vueprofile" on HP/UX, determine what type of machine is running and set the CEKEYS environment variable to the appropriate name. For example:

"export CEKEYS=~/.Cekeys.hp"

If you are upgrading to a new release of ***Ce*** and would like the new key definitions, but do not want to loose your existing customizations, execute the following UNIX command:

UNIX: ce_update

This command will make add any new default key definitions to the end of your current ".Cekeys" file. Only keys which are currently undefined are added.

NOTE: The key definitions are held in your HOME directory in the file ".Cekeys". The definitions set up by "ce_init" are a basic editing suite. If you don't like the way a particular key behaves, you can probably change it to your liking. For example, The left arrow key, which moves the cursor one character left executes the Ce primitive command "al" and stops when it hits column 1 of the current line. Some people prefer to have the cursor jump up to the last character of the previous line. Changing the key definition to execute "al –w" (arrow left with wrap) will do this. If you can't see how to do something, write for help via email to the support address.

Creation of a *Ce* Session Using a Practice File

Enter the following command from a shell prompt:

UNIX: ce ~/ce_practice

The file "$HOME/ce_practice" was created by "ce_init" If you receive the message "Unable to find key definition file", you will still get a ***Ce*** session; however, your keys will not be properly initialized. This should not happen if you executed the "ce_init" command prior to running *ce*. ***Ce*** accepts a number of command line options. Those options are discussed in *chapter 3, Environment Customization*. See *Initialization of Your Personal Ce Environment* Section.

The Appearance of a *Ce* Session on Your Display

Figure1-1 shows the basic components of a ***Ce*** session. (Since ***Ce*** runs in a number of X Window System environments, such as OSF/Motif and Open Windows, the Figure does not describe details that relate to the window manager.) The functions of the various components are as follows (the numbers of these descriptions correspond to the circled numbers in the illustration):

1. This is the name of the file that you are editing. The entire pathname for the file is shown whenever possible.

2. If you are running ***Ce*** on one system and displaying its window on another, the name of the system on which ***Ce*** is running is shown here.

3. This is known as the window's title bar. It contains the file name (1), one or more edit mode indicators (5), and the number of the first visible line in the window (7).

4. This is the column indicator, present if the window is scrolled at all to the right. The number shown indicates which physical file column is currently visible in the window's left–most display column. It only appears if the left-most column displayed is not one.

5. This is an edit mode indicator. The "I" indicates that the file is in insert mode. If the file is in overstrike mode, the "I" is not present. If the file is in read–only mode, an "R" appears at this location; also, a file in read–only mode cannot be in insert or overstrike mode, so the "I" will never appear when the "R" is present.

6. This is the modified indicator. The presence of the "M" means that the file has been modified sometime after the ce session was started. The flag is not present when the copy of the file in memory matches what is on disk.

7. This is the top line number. It indicates the number of the first line that is visible in the window.

8. This is the line number window. It shows the physical line numbers for the text lines that appear in the window. This is a read–only window, and it scrolls automatically whenever the text window (9) is scrolled. It is toggled into existence by the "lineno" command.

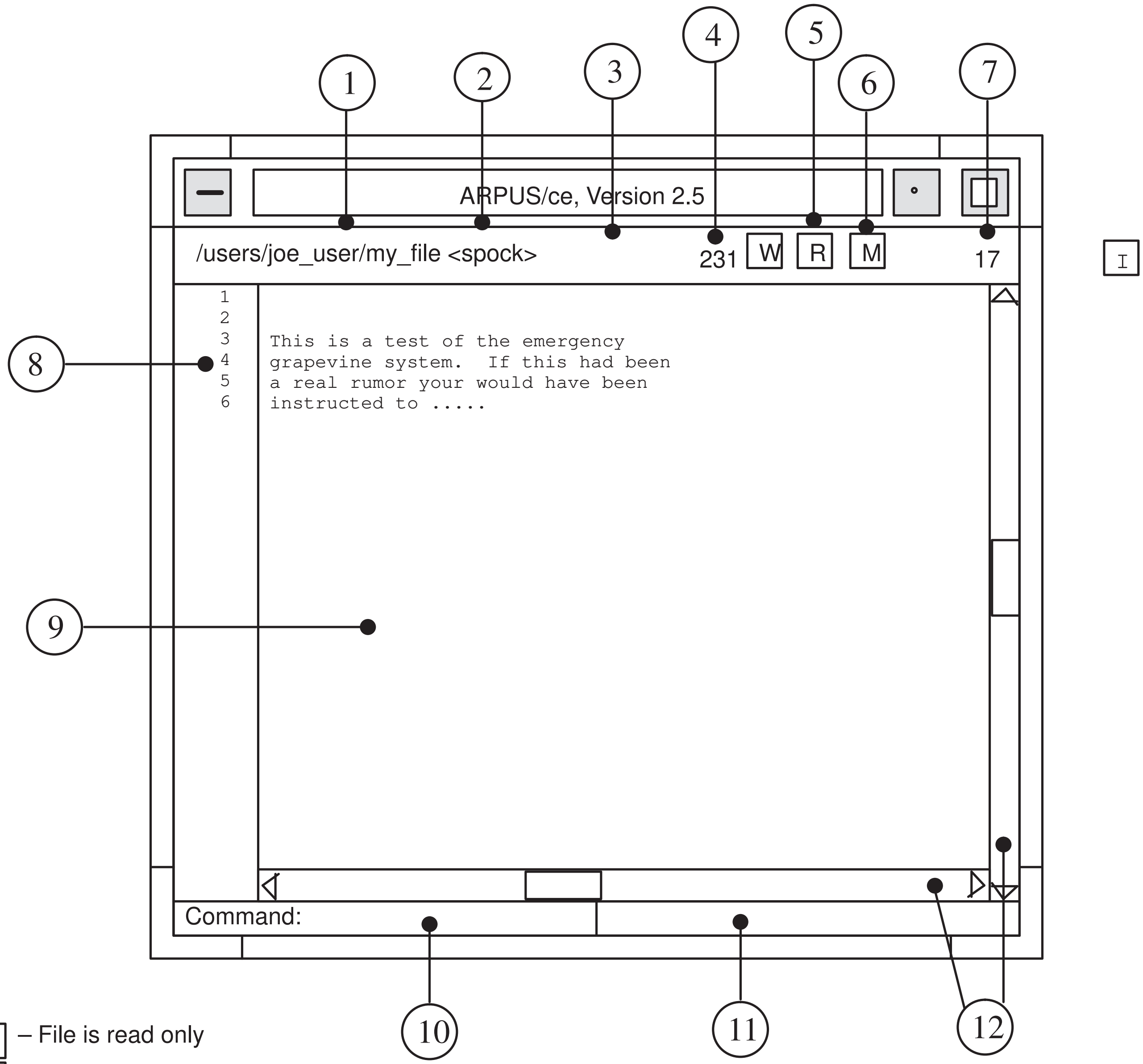
9. This is the text window. It contains the text that you are editing or viewing.

10. This is the command input window. You use this window to enter various edit commands, as will be demonstrated in the section "Search and Search/Replace Operations".

11. This is the command output window. ***Ce*** uses this window to display command results or error messages. It is a read–only window.

12. These are the scroll bars. They can be set to turn on as needed, always be on, or never be on. They are controlled with the sb command and/or the scrollBar X resource.

Note that when you place the mouse cursor in either (9), (10), or (11), the mouse cursor and text cursor stay together, unless the "–mouse no" command line option or the "Ce.mouse: no" X resource are used.



R – File is read only

W – Word wrap mode (See help for WW command)

I – Insert mode, same position as R

M – File has been modified since last save

NOTE: The scroll bars can be toggled on and off with the “sb” command and the line number column can be toggled on and off with the “lineno” command. The “lineno” and “scrollBar” X resources can be used to set the initial state of line numbers and the scroll bars.

Figure 1-1. Appearance of *Ce* Session

Cursor Control and Window Scrolling

Cursor control involves moving the text cursor within the window. Table1-1 lists the cursor movements that are possible, along with the keys that cause those movements. The key names shown in Table 1-1 are keys that are common to all workstation keyboards. Moving the mouse also moves the cursor.

Table 1-1. Cursor Control

Movement	Default key(s)
Cursor up	Arrow Up
Cursor down	Arrow Down
Cursor left	Arrow Left
Cursor right	Arrow Right
Tab right one stop	Tab
Tab left one stop	<Shift> Tab
To start of line	Home
To end of line	End
To “Command: ” prompt	Escape

Window scrolling is accomplished via keypresses or the scroll bar. By default, the vertical and horizontal scroll bars appear as necessary. The window can be scrolled either vertically (up and down by lines or pages) or horizontally (left and right by columns). Table 2.2 lists the scroll operations and their default key assignments.

Table 1-2. Window Scrolling

Movement	Default key(s)
To top of file	<Ctrl> t
To bottom of file	<Ctrl> b
Page up 1/2 page	Page Up
Page down 1/2 page	Page Down
Page up 1 line	<Shift> Arrow Up
Page down 1 line	<Shift> Arrow Down
Scroll Right 10 Columns	<Shift> Arrow Right
Scroll Left 10 Columns	<Shift> Arrow Left

Do not be concerned if the above key function assignments are not to your liking. *Chapter 8, Key Customization* provides detailed information on modifying these definitions as well as creating your own.

Basic Text Editing Operations

In its default operating mode, ***Ce*** links the mouse and text cursors. Thus entering alpha–numeric and other printable characters requires only placing the cursor at the proper point of insertion and typing the desired characters.

Table1-3 lists the most basic editing operations. Some keyboards, such as the SUN type-5 keyboard or the HP Workstation keyboard, have special extended function keys. These are mapped as appropriate.

Table 1-3. Basic Editing

Operation	Default key	Extended key
Delete character under cursor	Delete	
Delete character to left of cursor	Backspace	
Delete entire line under cursor	F5	Del
Delete from cursor to end of line	<Shift> F5	
Delete word under cursor	<Ctrl> F5	
Insert most recently cut line	F6	Insert
Insert most recently cut to–end–of–line	<Shift> F6	
Insert most recently cut word	<Ctrl> F6	
Undo most recent change	F7	Undo
Redo most recent undo	<Shift> F7	Redo
Split the current line	Return	Enter
Insert a new line below cursor	<Shift> Return	
Toggle between insert and overstrike	Insert	

Modes

As mentioned earlier, ***Ce*** operates in two distinct modes with respect to text editing: insert mode and overstrike mode.

While in insert mode

- characters under and to the right of the cursor are shifted to the right to make room for new characters being typed

While in overstrike mode

- Characters under the cursor are replaced by the new characters being typed
- “Backspace” causes the character to the left of the cursor to be replaced by a space
- “Return”, when entered with the cursor in the middle of the line, causes the line to split, the character under the cursor to be deleted, and all text to the right of the cursor is moved to a new line.

The differences between insert and overstrike modes are obvious when entering straight text but are somewhat more subtle with respect to Backspace and Return operations.

Undo and Redo

Note that any change, including those mentioned in Table 1-3, can be undone via the “undo” command, which is assigned to the F7 key. Furthermore, if you undo a change and then decide that you really did want that change, you can undo an undo via the “redo” command, which is assigned to the “<Shift> F7” key combination. ***Ce*** allows you to undo all the way back to the beginning of the edit session. If you undo a change and make a different change, you cannot redo the original change you undid.

Non–Rectangular Copy/Cut/Paste Operations

Ce provides easy ways to duplicate portions of text and move portions of text from one location in the file to another. Such operations are facilitated through the use of paste buffers. Copying text, moving text, and deleting (cutting) text all involve the same basic steps, and by default these steps are accomplished using the function keys F1, F2, F3, and F4. Table 1-4 shows the steps required to perform any of those three operations.

Table 1-4. Normal (non–Rectangular)Copy/Moving/Cutting Operations

	Copying text	Moving text	Cutting text
Step 1	Mark start of text range via F1	Mark start of text range via F1	Mark start of text range via F1
Step 2	Move cursor to end of text range	Move cursor to end of text range	Move cursor to end of text range
Step 3	Copy text via F2	Cut text via F3	Cut text via F3
Step 4	Reposition cursor at copy's location	Reposition cursor at text's new location	
Step 5	Paste text via F4	Paste text via F4	

In all cases, Step 3 places the copied or deleted text into a paste buffer, which is a separate entity that is associated with but not an actual part of the file you are editing. This means that ***Ce*** remembers pieces of text that you copy or delete so that if and when you subsequently issue a paste command, ***Ce*** knows what to paste back into the file.

When defining a text region to copy or cut, the default behavior of the operation is that all text between the marked and final cursor positions is included in the region, as shown in Figure 1-2.

```
This illustration shows highlighted
text for a normal
(that is, non–rectangular) copy or
cut  operation.  Note  that  the
highlighted region  includes all text
between the marked and current cursor
positions, as if the file were one
long text line that wraps around onto
numerous lines (which is in fact what
it really is).
```

Figure 1-2. Normal (non–Rectangular) Copy/Cut Region

The lines that end up in the paste buffer after a copy or cut of the region highlighted in Figure 1–2 would look like this:

```
Note  that  the
highlighted region  includes all text
between the marked and current cursor
positions,
```

Rectangular Copying, Cutting, and Pasting.

You also have the ability to handle rectangular blocks of text. The mark, copy, cut, and paste operations all have rectangular analogs, initiated via the use of shifted function keys, as Table 2.5 shows:

Table 1-5. Rectangular Copy/Moving/Cutting Operations

Operation	Normal Key	Actual *Ce* Command	Rectangular Key	Actual *Ce* Command
Mark	F1	dr; echo	<Shift> F1	dr ; echo –r
Copy	F2	xc	<Shift> F2	xc –r
Cut	F3	xd	<Shift> F3	xd –r
Paste	F4	xp	<Shift> F4	xp –r

Figure 1-3 and the paragraph immediately following it illustrate the appearance of a highlighted rectangular text region and the lines in a paste buffer copied or cut from such a region.

```
This illustration shows highlighted
text for a normal
(that is, non-rectangular) copy or
cut  operation.  Note  that  the
highlighted region  includes all text
between the marked and current cursor
positions, as if the file were one
long text line that wraps around onto
numerous lines (which is in fact what
it really is).
```

Figure 1-3. Rectangular Copy/Cut Region

The lines that end up in the paste buffer after a copy or cut of the rectangular region highlighted in Figure 1-3 would look like this:

```
highlig
between
positio
long te
```

NOTE: Be aware of this important characteristic about the differences between normal and rectangular operations: Once the text is in the paste buffer, no record is kept about whether the operation that put the text there was normal or rectangular. If you want to do a rectangular copy followed by a rectangular paste, you must use the rectangular versions of both commands.

The Default Paste Buffer vs. Named Paste Buffers

By default, the copy, cut, and paste commands all use the same internal paste buffer. As a result, a copy or cut operation wipes out any data in the paste buffer that may have been placed there from a previous copy or cut operation.

Similarly, once you perform a copy or cut operation, that data can then be pasted repeatedly back into the file until another copy or cut operation is performed.

For most applications, the above behavior is acceptable. However, there may be instances when you would like to perform two or more copies or cuts and preserve the data from each of those operations in a distinctive fashion. ***Ce*** allows you to create named paste buffers. The default key definitions that you are using for F1, F2, F3, and F4 have equivalent functions <Ctrl> F1, <Ctrl> F2, <Ctrl> F3, and <Ctrl> F4 which use an alternative paste buffer. Setting up additional paste buffers is beyond the scope of this chapter. The subject is handled in detail in the *Chapter 3, Advanced Editor Use*.

Cut and Paste between Ce and other X based applications

It is often desirable to copy and paste between a ***Ce*** window and another application. The desired behavior would be to do a normal copy in the ***Ce*** window, move to the other application, and perform the paste using the method supplied by that application. A paste pulldown or pressing the middle mouse button are two common methods. The trick is to make sure you are using the same paste buffer name in both applications. There are two names commonly used by X based applications. They are named “PRIMARY” and “CLIPBOARD”. Which name is used by any application is subject to change, but a good rule of thumb is if you are pasting with the middle mouse button, it is probably pasting “PRIMARY” and if you are using a pulldown, it is probably “CLIPBOARD”. If a particular instance does not work, try the other one.

Since Ce allows you to vary the paste buffer names, it is generally easier to adjust Ce to talk to your other applications than the other way around. There are two ways of doing this. One is to use the “Ce.dfltPasteBuf X” resource. By default this is set to the name “CLIPBOARD”. However, this may be set at installation time. Use the UNIX command “ce –oplist” to see if a new value was specified at installation time. To change this to “PRIMARY”, put the line, “Ce.dfltPasteBuf : PRIMARY”, In your “.Xdefaults” file. Changing this resource changes the paste buffer name used on copy “xc”, cut “xd”, and paste “xp” ***Ce*** commands which do not explicitly name a paste buffer. For systems other than SunOS 4.1.*, it is probably wise to change the default paste buffer to “PRIMARY”.

Search and Search/Replace Operations

You can instruct ***Ce*** to search for an exact sequence of characters or a sequence of characters that match a given pattern. Furthermore, you can define a region and instruct ***Ce*** to replace all occurrences of a character sequence or pattern within that region with another character sequence.

Search Operations

To search forward in the file for a specific string, do the following:

1. Place the cursor at the point where you want to begin the search.

2. Press the “Escape” key, which will move the cursor to the “Command:” prompt.

3. Type “/string/” (not the quotes, just the two slashes with the search string between the slashes), and press “Return”.

If the search succeeds, ***Ce*** will place the cursor at the beginning of the occurrence of “string”, and it will scroll the window if necessary to make visible the line on which the occurrence was located.

If the search fails, ***Ce*** will indicate so by displaying the message “No match: string” in the command output window, and the cursor will be left in its original position.

If you want to search for “string” repeatedly, you need perform the above command sequence only the first time. You can repeat a forward search by pressing “<Ctrl> r”, which translates to the Ce command “//”.

Search operations may take into account whether the string being searched for and the text being searched are upper or lower case. Searches can happen in one of two modes:

- Case insensitive - Upper case characters and lower case characters (e.g., “z” and “Z”) are considered equivalent, so that, for example, the command “/find me/” would match the text “Find Me”. This is the default search mode.
- Case sensitive - Uppercase characters and lowercase characters are treated distinctly, so that, for example, the command “/hello/” would not match the text “Hello”. This may be changed dynamically with the “sc” command. The default may be changed with the “Ce.caseSensitive X” resource.

You can also search backward in a file for a string. The proper form of the backward search command is “\string\”. That is, enclose the string to be searched for between backslashes. Likewise, to repeat a backward search, press CTL/u, which translates to the ***Ce*** command “\\”. The command “?string?” is equivalent to “\string\”. Also, the trailing slash or question mark may be omitted.

Search/Replace Operations

You can replace occurrences of a character sequence or pattern with another within a defined region of text. The steps required for a search-and-replace operation are as follows:

1. Place the cursor at the beginning of the desired region and press “F1” (Mark).
2. Move the cursor to the end of the desired region, and then press “Escape”, which will put the cursor at the “Command:” prompt.
3. Type “s/search_string/replacement_string/”, then press “Return”.

All occurrences of “search_string” will be replaced by “replacement_string” within the defined region. If no occurrences of “search_string” are found, the message “No match” will be displayed in the command output window and the text highlight will be aborted.

A second search–and–replace command is “so”, which instructs ***Ce*** to replace only the first occurrence of “search_string” on each line in the defined region.

The formats for “search_string” and “replacement_string” can include wildcard characters and other pattern matching mechanisms. See *Chapter 3, Advanced Editor Use* for details on such forms of search-and-replace operations. The slash character may be replaced with an alternate delimiter of your choice if the search or replace straings contain slashes. You could alternatively escape using slashes in the search or replace strings.

Window Geometry

Normally you control window size and placement with the window manager. If you want to resize or relocate the ***Ce*** window during your edit session from within ***Ce***, you can use the “geo” command. The syntax for the command is “geo [c]WxH+x+y”, where:

- “W” specifies the window width and “H” specifies the height. “c” (the literal character “c”) is optional. If “c” is present, “W” and “H” indicate the number of columns and rows in the window; otherwise they indicate the number of pixels; and,
- “x” and “y” specify the point at which the upper–left corner of the window is located relative to the upper–left corner of the screen; “x” and “y” are always in units of pixels.

It is also possible to specify a default window geometry in your “$HOME/.Xdefaults” file or when ***Ce*** is invoked from the command line. It is also possible to have ***Ce*** rotate through a series of initial window placements, sizes, and color schemes. These additional methods are described in *Chapter 2, Environment Customization*.

On–line Help

Ce includes an extensive collection of on–line help files that are available through the click of a button:

- F9 - If you press “F9”, ***Ce*** will prompt you for a command or topic on which you would like help. At the prompt, you can enter a command name (such as “xc”) or a more general topic (such as “patterns”). A null response (just press return) will display an introductory help topic on using help. On Sun workstations, the key marked “Help” also works. On some HP keyboards, F9 is the leftmost blank key in the set of 4 blank keys in the upper right corner of the keyboard.
- M3 - (Right Mouse button) Most help files conclude with a reference section that lists additional help files for related commands and topics. The references indicate the exact system path names where the help files are located. If you place the cursor on one of those path names and press “M3”, ***Ce*** will create a window with that help file in it.

If you are having trouble finding help on a topic, press “F9”, and look at the topic “commands”, which lists all the ***Ce*** commands. The HTML documentation has a complete index which may also be useful.

Bug Reports, Questions, and Enhancement Requests

We are interested in continuously providing you with high–quality software and support, and are interested in information you have about problems with the software and also in your ideas on how the software can be enhanced.

When you want to send information about ***Ce*** to ETG, use the shell command “ce_report”. (This is a shell command and not a ***Ce*** edit command.) “ce_report” copies into your home directory a pre–formatted file into which you can type information relevant to your problem or enhancement request. The file also contains specific information on how to send the completed file to ETG via surface mail, electronic mail, or fax.

We also encourage you to take advantage of our telephone support at
(770) 64–ARPUS (770) 642–7787

Chapter 2: Environment Customization

Organization of This Chapter

This chapter will explain how you can control characteristics of your edit sessions by way of command line options and X resource specifications.

This chapter describes the following:

- Command line options and X resource specifications
- The $HOME/.Xdefaults file
- Option precedences
- The “cv” command
- Ce resources vs. ce, ceterm, and cv resources
- Option descriptions

Command Line Options and X Resource Specifications

On many systems, X resource specifications are stored in your “$HOME/.Xdefaults” file. As the table below shows, most options can be set both as command line options and X resources, while others can be set only in one way. The installation of ***Ce*** creates “file /usr/lib/X11/app–defaults/Ce” or the file Ce in a directory of the installers choosing. This file may be inspected to see if any of the default values have been overridden for your site. You may also execute the UNIX command, “ce –oplist” to see modified values.

Table 2-1. Command Line Options and X Resource Specifications

Command Line Form	Xdefaults Form	Default Setting
–autoclose {y|n}	Ce.autoclose: {y|n}	no autoclose
–autohold {y|n}	Ce.autohold: {y|n}	no autohold
–autosave <number>	Ce.autosave: <number>	no autosave
–background <color>	Ce.background: <color>	white
–bkuptype {dm|vi|none}	Ce.bkuptype: {dm|vi|none}	dm
	Ce.cekeys: <file>	$HOME/.Cekeys
–cmd “cmd[;...]”	Ce.cmd: cmd[;...]	no commands
–display <display>	–	$DISPLAY
–dotmode {0|1[string]|2}	ceterm.dotmode:<name>	–
–dpb <name>	Ce.dfltPasteBuf:{0|1[string]|2}	clipboard

Table 2-1. Command Line Options and X Resource Specifications (Cont)

Command Line Form	Xdefaults Form	Default Setting
–envar {y|n}	Ce.ebvar:{y|n}	yes
–expr {aegis|unix}	Ce.expr: {aegis|unix}	aegis
–findbrdr <num>	Ce.findbrdr: <num>	0
–font <font>	Ce.font: <font>	fixed
–foreground <color>	Ce.foreground: <color>	black
–geometry <geometry>	Ce.geometry: <geometry>	640x512+0+0
–help	–	–
–ib	Ce.iconBitMap: <filepath>	internal
–iconic	Ce.iconic:{y|n}	no
–isolatin {y|n}	Ce.isolatin: y|n}	no isolatin
–kdp <name>	Ce.keydefProp: <name>	CeKeys
–lineno {y|n}	Ce.lineno: {y|n}	no line numbers
–linemax <number>	ceterm.linemax: <number>	no column limit
–load	–	load only on first use
–lockf {y|n}	Ce.lockf: {y|n}	yes
–ls	ceterm.loginShell:{y|n}	no
–LSHOST <nodename>	Ce.LSHOST: <nodename>	–
–man	–	no “man” processing
–mouse {on|off}	Ce.mouse: {on|off}	on
–name <name>	–	command name
–nb	–	–
–noreadahead	Ce.noreadahead: {y|n}	no
–offdspl {y|n}	Ce.offdspl: {y|n}	no
–oplist	–	–
–pbd <dir>	Ce.pasteBuffDir: <dir>	~/.CePaste
–pdm {y|n}	Ce.pdm: {y|n}	yes
–reload	–	no reload

Table 2-1. Command Line Options and X Resource Specifications (Cont)

Command Line Form	Xdefaults Form	Default Setting
–readlock	–	not locked
–sb {y|n|auto}	Ce.scrollBar: {y|n|auto}	auto
–sbcolors'g hl sl sh'	Ce.scrollBarColors: g hl sl sh	–
–sbwidth <num>	Ce.scrollBarWidth: <num>	256
–sc {y|n}	Ce.caseSensitive: {y|n}	no
–scroll {n|y}	Ce.scroll: {y|n}	no
–stdin	–	not active
–tabstops "ts cmd"	Ce.tabstops: ts_cmd	
–tbf <name>	Ce.titlebarfont: <name>	fixed
–title "string"	Ce.title: title string	ARPUS/Ce
–transpad	–	not active
–ut	Ce.utmpInhibit: {True|False}	–
–version	–	–
–w		no waiting

Descriptions for each of the above options can be found in the *Option Descriptions Section* of this Chapter.

The $HOME/.Xdefaults File

After installation of ARPUS/Ce, the app–defaults file "Ce" file contains a template that lists the above resources along with their descriptions. All of the***Ce*** resources in that template are disabled except for "Ce.CEHELPDIR" and ones enabled as site wide defaults by the system administrator. To enable a desired resource setting in that file, copy the line to your ".Xdefaults" file and remove all leading '!' characters from the beginning of line. The default location for the ***Ce*** app–defaults file is "/usr/lib/X11/app–defaults". The person peforming the installation of ***Ce*** may choose to place the file elsewhere. See *Appendix E, Installation Summary* for more details if you are this person. If the file is not in the directory "/usr/lib/X11/app–defaults" you can check each of the directories in the environment variable XFILESEARCHPATH for a file is named "Ce" (upper case 'C' lower case "e"). You can view the contents of this variable from any shell (sh, ksh, or csh) by executing the UNIX command "echo $XFILESEARCHPATH". If you, and especially ***Ce***, cannot find the app–defaults file, ***Ce*** will pick up its internal defaults. Typing "ce –help" will produce a list of all the options and a one line description of each.

Once you determine which options you want to change, add them to your "$HOME/.Xdefaults" file and place the desired values in place. These will become the defaults for your account.

You can view the options ***Ce*** is using (assuming no command line options) by executing the command “ce –oplist”.

NOTE: Anytime you modify the resource specifications in your “$HOME/.Xdefaults” file, you must run the command “xrdb –merge $HOME/.Xdefaults” to update the X server with the new settings. The location of “xrdb” varies from system to system. If you have difficulty locating it, contact your system administrator.

Option Precedence

The following rules determine how resources settings are determined:

- If neither the command line form nor the “.Xdefaults” form are used, the default setting takes effect.
- If the command line form is used, the specified setting takes effect for that edit session only.
- If only the “.Xdefaults” form is used, the specified setting takes effect for every edit session.
- If both forms are used, the command line setting overrides the “.Xdefaults” setting for that edit session only.

The “cv” Command

The “cv” command behaves in exactly the same way as “ce”, except that when the edit window is created it is initially in read–only mode. This is a handy way of browsing through a file without risking accidental changes to that file, since changes are not allowed in a read–only window. The “cv” window can be converted to a ce window with the “ro” ***Ce*** command, the “<ctrl>–m” key sequence, or the “read–write” option in the “Modes” pulldown. You cannot convert a “cv” to a “ce” unless you have write access to the file.

Ce Resources vs. ce, ceterm, and cv Resources

The “.Xdefault” forms of the options shown in Table 2.1 take effect for “ceterm”, “cv”***,*** and “ce” windows. However, if you would like one set of options to apply to “ceterm” windows and another set *ce* windows, you can vary the .Xdefault option prefixes as follows:

- For an option to apply to all three, use “Ce.”, e.g., “Ce.lineno: yes”.
- For an option to apply to ce only, use “ce.”, e.g., “ce.lineno: yes”.
- For an option to apply to cv only, use “cv.”, e.g., “cv.lineno: no”.
- For an option to apply to ceterm only, use “ceterm.”, e.g., “ceterm.lineno: no”.

Case is important — ce and cv recognize the difference between “Ce” and “ce” and ce takes precedence over Ce.

Option Descriptions

The .Xdefaults forms below all show the “Ce.” prefix for the sake of brevity. Unless otherwise specified, “Ce.” can be replaced by “ce.” or “cv.” to make the option apply specifically to “ce” or “cv“ respectively. Also, ***Ce*** recognizes unique abbreviations of the command line forms; those abbreviations are underlined. For your convenience, these option descriptions also appear in the on–line help file titled “Xresources”.

–autoclose {y|n}
Ce.autoclose: {y|n}

If this option is enabled, ***Ce*** will automatically close a “ceterm“ window once its corresponding shell process is terminated.

–autocut {y|n}
Ce.autocut: {y|n}

This option, when set to “yes“, causes the “ed“ (delete), “ee“ (backspace), “es“ (type a character), “er“ (type a special hex character), and “xp“ (paste) commands to automatically cut a highlighted area when they are in a key definition by themselves. That is, not part of a list of Ce commands executed under one keystroke. This is similar to Netscape and other PC based editors.

–autohold {y|n}
Ce.autohold: {y|n}

If this option is enabled, ***Ce*** will automatically place a “ceterm” window in hold mode when a window full of output is received by the window.

–autosave n
Ce.autosave: n

If this option is enabled, ***Ce*** will automatically update the file changes to disk after every n keystrokes which modify the file.

–background color
–bg color

Ce.background: color

Set the background color to color upon invocation. “color” must be a valid color found in the X server’s “rgb.txt“ file. (The directory in which that file resides varies between platforms and is probably one of “/usr/X11/lib“, “/usr/lpp/lib/X11/lib”, “/usr/lib/X11”, or “/usr/lib/X11/ncd.)”. Note that on Apollo Domain/Os systems, “color” must appear in double quotes. If you are going to use the ”wdc” command in your “.Cekeys” file to rotate through a set of colors, explicitly set background color to a null value, eg. “Ce.background:“.

–<u>bk</u>uptype {dm|vi|none}
Ce.bkuptype: {dm|vi|none}

Set the mode in which ***Ce*** backup files are created. “–nb” is a valid abbreviation for “–bkuptype none”.

In “dm” mode, the original file edited is renamed to “file.bak” and a new file is created. Note that this means that any hard links that originally pointed to original file will subsequently point to “file.bak”.

In “vi” mode, file is copied to file.bak. This means that hard links will still be valid, pointing to the changed version of file.

Use “none” if backup files are not desired.

Ce.cekeys : <path>

Path to file containing site wide key definitions. During key definition loading, ***Ce*** does an implied include (cmdf) of this file to pick up site wide definitions. The users personal “.Cekeys” file can override these. The lookup is actually a search. First a check is made for the specified file with either a “.A” or “.U” attached depending upon whether the user is in Aegis regular expression mode (.expr resource) or UNIX regular expression mode. If that file does not exist, the name as specified in the file is checked for. This allows for expression mode dependent key definitions. You can of course include (via cmdf) the base file from the expression mode specific files. DEFAULT: Set at install time.

–<u>cm</u>d “cmd[;...]”
Ce.expr: cmd[;...]

When ***Ce*** starts up, it will automatically execute the specified commands before giving control to the user.

–<u>di</u>splay display

Use the specified display. This must be a valid X host display specification. If the display is not specified via the command line or the “.Xdefaults” resource, the X server selects the display based on the value of the “$DISPLAY” environment variable.

–<u>envar</u> {y | n}
Ce.envar: {y | n}

This parameter controls the processing of environment variables found in path names for the “cv” and ***Ce*** commands executed in the “Command:” window. When set to yes, clicking on a name such as “$HOME/.profile” will expand “$HOME”. It is possible to escape the dollar sign on the name to avoid expansion if the “cv” command is typed in as opposed to being generated by clicking on the name. When set to no, dollar signs are ignored. This allows clicking on names which contain dollar signs.

–<u>ex</u>pr {aegis|unix}
Ce.expr: {aegis|unix}

Ce understands both AEGIS and UNIX regular expression syntax. See *Chapter 4, Advanced Use* for a detailed treatment of AEGIS and UNIX regular expressions.

–<u>fon</u>t font
Ce.font: font

Use the specified font, which must be a valid X font. Most systems provide the command “/usr/X11/bin/xlsfonts”, which produces a list of fonts available on the host system. The use of variable–width fonts is discouraged.

–foreground color
–fg color
Ce.foreground: color

Set the foreground color to color upon invocation. The “color” must be a valid color found in the X server’s “rgb.txt” file. (The directory in which that file resides varies between platforms and is probably one of “/usr/X11/lib”, “/usr/lpp/lib/X11/lib”, “/usr/lib/X11”, or “/usr/lib/X11/ncd”.) Note that on Apollo Domain/Os systems, “color” must appear in double quotes. If you are going to use the “wdc” command in your “.Cekeys” file to rotate through a set of colors, explicitly set foreground color to a null value eg. “Ce.foreground:”.

–geometry [c]WxH+x+y
Ce.geometry: [c]WxH+x+y

Use the specified geometry when sizing and locating the window. The geometry must be a valid X geometry specification. The leading “c” is optional; if specified, ***Ce*** interprets the width and height geometry values as character columns and rows instead of x and y pixels. If you are going to use the “wdf” command in your “.Cekeys” file to rotate through a set of window geometries, explicitly set geometry to a null value, eg. “Ce.geometry:”.

–help

This option causes ***Ce*** to print a summary of the command line options and exit. This option is not available from the “Command:” prompt.

–iconic

Start ***Ce*** as an icon. This option is useful in startup scripts (such as the .xinitrc fille) to start some ce or ceterm windows as an icon.

–isolatin {y|n}
Ce.isolatin: {y|n}

If this option is set to “yes”, ***Ce*** will interpret certain keystrokes as isolatin character representations.

–lineno
Ce.lineno: {no|yes}

If this is set to “yes”, file line numbers are shown along the left margin. If you include “–lineno” on the command line, ***Ce*** forces the line number column to appear regardless of the X resource setting or the default setting.

–load

When ***Ce*** is invoked during a new login session, it must parse your key definitions file and store the definitions in the X server database. Files containing many key definitions may take a few seconds to process. You can avoid that delay by putting the command “ce –load” in your “.xinitrc”, “.dtprofile”, of “.vueprofile” startup fle as appropriate. The “–load” option tells ***Ce*** to process your key definitions file, store the definitions in the X server database, and terminate without opening an edit window. Subsequent ***Ce*** commands will create the edit window more quickly since the key definitions are already loaded.

–linemax
Ce.linemax: <num>

This option can cause Ceterm to limit the number of lines it retains in the transcript pad. By default, the upper limit is approximately 16 million lines.

–ls
Ce.loginShell: {y|n}

This option indicates that the shell that is started in the ceterm window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's ".login" or ".profile").

Ce.LSHOST: <node_name>

This option assists in the use of the license server. The license server client code checks the environment variable "LSHOST" to find the node running the license manager. If the environment variable is NOT set in the environment at the time ***Ce*** is started, the value in this resource is placed in the environment for use by the license server client code. This option is usually set in the "app–defaults" file for ***Ce*** by the system administrator. Note that if the "LSHOST" variable is already set in the environment, this value is ignored.

–man
ceterm.man: {y|n}

Numerous systems have "man" pages that contain embedded backspaces for the sake of underlining and other expressive text constructs. ***Ce*** normally does not process backspace characters specially, making files that contain them difficult to read. You can use the "–man" option to tell ***Ce*** to eliminate those backspace characters and the accompanying "overstrike" characters to make the file more readable. This option also causes ***Ce*** to ignore null characters (hexadecimal value 00) in the file being viewed. For ceterm, this option indicates that vt100 control sequences sent to the terminal by the shell when the ceterm is not in vt100 emulation mode should be ignored.

–mouse {y|n}
Ce.mouse: {y|n}

You can use this option to indicate that the mouse cursor and text cursor should follow each other. If you set this to "no", the cursor move independently; the "sic" command can then be used to move the text cursor to the mouse cursor's current position. By default the M1 key is defined as "kd m1 sic ke".

–noreadahead
Ce.noreadahead {<u>no</u>|yes}

Normally, ***Ce*** reads in enough data from the file being edited to display the first screen, displays the screen, and spends the time in between user actions reading the file in a background mode. The "–noreadahead" parameter instructs ***Ce*** not to do this and only read in data as it is needed. This is useful when looking at the top few lines of very large files.

–<u>pb</u>d directory
Ce.pasteBuffDir: directory

Use the specified directory to hold paste buffer files. Note that the default paste buffer directory resides in "$HOME.CePaste".

–<u>rea</u>dlock

This option tells ***Ce*** to operate in read only mode and disables the ro command. Thus you cannot toggle out of read only mode in that window. This feature is used by programs which invoke ***Ce*** and do not want the user to enter edit mode.

–<u>re</u>load

This option tells ***Ce*** to reload your key definitions into the X server database even if the database already contains your key definitions. This option forces the X server to accept a new set of key definitions and is useful if you are experiencing problems with the X server.

–sc
Ce.sc: {no|yes}

Set case sensitivity for searches. Once ***Ce*** is started, case sensitivity can be changed via the “sc” command. If you include “–sc” on the command line, ***Ce*** forces case sensitivity on regardless of the X resource setting or the default setting.

–scroll: {y|n}
Ce.scroll: {n|y}

If set to “yes”, a ceterm window is scrolled one line at a time as output is produced by commands running in the ceterm session. If set to “no”, the output is scrolled one windowfull at a time.

–ts: “ts [s1 [s2 ... sn] [–r]]”
Ce.tabstops: ts [s1 [s2 ... sn] [–r]]

Set tab stops in columns s1, s2, sn. If “–r” is present, add additional stops spaced according to the difference between the last two specified stops.

–title: “<string>”
Ce.title: <string>

Set the ***Ce*** session’s Window Manager window title to the specified string. If this is option is not used, the default window title is “ARPUS/ce Version 2.5”.

–ut
Ce.utmpInhibit: {True | False}

This option performs the same function as the xterm “–ut option”. It affects ceterm only. Normal operation calls for ceterm to generate a Unix utmp entry which is viewable via the Unix “who(1)” command. Specifying this option inhibits the generation of the utmp entry. On some platforms, specifying “–ut” allows a ceterm to be started without the program having special setuid privileges.

–version

This option causes ***Ce*** to print its version identification and terminate.

–vt: {y|n|auto}
ceterm.vt: {y|n|auto}

This option applies only to ceterm windows. If set to “auto” ceterm will automatically enter vt100 emulation mode whenever the shell requests that the “echo” line discipline is disabled. This more accurately reflects the behavior of the terminal windows on an Apollo Domain system. Use of this feature will cause ceterm to enter vt100 mode whenever a password is being requested and whenever a telnet or rlogin is performed. The value “auto” disables the toggle action of the “vt” command. The value “on” will cause ceterm to start in vt100 emulation mode. The value “off” will cause ceterm to start in normal mode. The “vt” command may be used to change this state on the fly.

–w

Normally the ***Ce,*** cv***,*** and ceterm shell commands spawn a separate editor process, and the shell in which the command was entered returns immediately with the shell prompt. “–w” instructs the editor to not return control to the shell until you close the edit window. This allows you to use this editor inside of a mail program, for instance.

This page intentionally left blank

Chapter 3: Advanced Use

Organization of This Chapter

This chapter takes you through an advanced ***Ce*** edit session. We encourage you to follow the examples by using ***Ce*** while you read this chapter.

This chapter describes the following:

- Regular Expressions
- Copy/cut/paste Operations
- Command Scripts
- File Saving and Backup Modes
- Tab and Form Feed Character Processing
- Editing the same document on multiple workstations
- Record / Playback Modes – Learn Mode

All of the information is also available on-line in the various ***Ce*** help files and HTML documentation. In many cases the online documentation is more complete with more examples than this users guide.

Regular Expressions

Regular expressions (also known as “patterns”) are used to find patterns of characters in a text region. The forward search, backward search, substitute, and substitute once commands all exploit regular expression functionality for performing sophisticated text searches and substitutions.

Ce supports the use of both AEGIS regular expressions and UNIX expressions (per the UNIX “ed” command). Only one style prevails per edit session, and that style can be explicitly set by either the the “–expr” command line option or the “Ce.expr” X resource. The default regular expression mode is AEGIS mode.

AEGIS regular expressions and UNIX regular expressions have much in common. Nonetheless the following descriptions treat the styles separately. This chapter does conclude with a table summarizing the similarities between the two styles.

AEGIS Regular Expressions

%	Matches the beginning–of–line, but only if it is the first character in the pattern; otherwise it is interpreted literally.
$	Matches the end–of–line, but only if it is the last character in the pattern; otherwise it is interpreted literally.
?	Matches any single character except newline.
*	Matches zero or more occurrences of the preceding character in the pattern, but only if this is NOT the first character in the expression; otherwise it is interpreted literally.
@f	Matches a form feed character (hex 0C).
@n	Matches a newline character (hex 0A).
@t	Matches a tab character (hex 09).
@c	Matches the literal character "c". This is used primarily to match a character that normally serves as a special regular expression character. For example, "@?" will match a literal "?".
[string]	Matches a single character that appears in "string". For example, "[abyz]" will match the first occurrence of either an "a", "b", "y", or "z". "string" may also indicate ranges of characters, so that "[a–cx–z4–6]" would match any one of "a", "6", "c", "x", "y", "z", "4", "5", or "6".
[~string]	Matches a single character that does not appear in "string". If "~" is not the first character following "[", it is interpreted literally. The same rules regarding ranges of characters that are defined in "[string]" also apply here.
{pat}	The curly brackets are used to "tag" an expression so that a string that matches the given pattern can be referenced in a substitution operation. For example, the substitution command "s/{[abc]}?/+@1+/" means that for every occurrence of an "a", "b", or "c" (i.e., "{[abc]}"), replace the occurrence with a "+" followed by the "a", "b", or "c" (i.e., "@1") followed by another "+".

All other characters appearing in AEGIS regular expressions are interpreted literally.

UNIX Regular Expressions

^	Matches the beginning–of–line, but only if it is the first character in the pattern; otherwise it is interpreted literally.
$	Matches the end–of–line, but only if it is the last character in the pattern; otherwise it is interpreted literally.
.	(period) Matches any single character except newline.
*	Matches zero or more occurrences of the preceding character in the pattern, but only if this is NOT the first character in the expression; otherwise it is interpreted literally.

\c	Matches the literal character "c". This is used primarily to match a character that normally serves as a special regular expression character. For example, "\." will match a literal ".".
[string]	Matches a single character that appears in "string". For example, "[abyz]" will match the first occurrence of either an "a", "b", "y", or "z". "string" may also indicate ranges of characters, so that "[a–cx–z4–6]" would match any one of "a", "b", "c", "x", "y", "z", "4", "5", or "6". To include a literal "~", "]", "–", or "\" in "string", precede the character with "\". If "]" is the first character in "string", it is interpreted literally and not as the pattern's terminating ']'.
[^string]	Matches a single character that does not appear in "string". If "^" is not the first character following "[", it is interpreted literally. The same rules regarding ranges of characters that are defined in "[string]" also apply here.
c \ { m \ }	Matches a string containing exactly "m" occurrences of the character "c", where "c" is either a literal character or a single character pattern (such as '.'). Similarly, "c \ { m , \ }" matches at least "m" characters, and "c \ { m , n \ }" matches between "m" and "n" characters. "m" and "n" must both be integers between 0 and 256. In this form of regular expression, the curly brackets and backslashes "\ { ...\ }" are required.
\(pat\)	The escaped parens are used to "tag" an expression so that a string that matches the given pattern can be referenced in a substitution operation. For example, the substitution command "s/\([abc]\)./+\1+/" means that for every occurrence of an "a", "b", or "c" (i.e., "\c[abc]\"), replace the occurrence with a "+" followed by the "a", "b", or "c" (i.e., "\1") followed by another "+".

All other characters appearing in UNIX regular expressions are interpreted literally.

Table 3-1. Similarities Between AEGIS and UNIX Regular Expressions

Meaning	AEGIS	UNIX
Beginning–of–line	%	^
End–of–line	$	$
Single character	?	.
Zero or more occurrences of the previous character in the pattern	*	*
Escape character to force literal interpretation	@	\
One character in a set	[string]	[string]
One character not in a set	[~string]	[^string]

Regular Expression Examples

Using regular expressions is a complex art and the subject of several text books. Following are some examples demonstrating various regular expressions in the context of find and substitute commands. Note these examples use UNIX regular expression mode.

kd Left \.\ ke

This key definition defines the left arrow key as searching backwards for any character. The backslashes indicate reverse search. The "." matches any character. Note that this definition creates a Left arrow key which jumps to the previous line when it gets to column one on a line.

1,$s/^ *//

Remove leading blanks from all lines in the file. The "1,$" is the range of lines. The "^" locks the pattern to the beginning of the line. "*" specifies zero or more of the preceding characters which is a blank. The replace string in the substitute is empty.

1,$s/ *$//

Remove trailing blanks from all lines in the file. The "1,$" is the range of lines. The "$" specifies that the blanks must be the last thing on the line. There are two spaces and a "*". The first space specifies that there must be one blank to match. The second space, followed by the "*" matches zero or more additional blanks.

s/ *\[/='%./

Over the following data:
printf("loadid   [8];   /* c */
printf("module   [31]; /* c */
" *\[" matches the blanks following "loadid" and "module" and the open square bracket. Note that the square bracket is escaped with a backslash to prevent it's being interpreted as a range of characters. The replacement characters "='%." are just literal characters.

1,$s/^/echo /

Put the word echo at the front of each line in a file. The "1,$" is the range of lines. The "^" matches zero characters at the beginning of the line. The word "echo" followed by a blank is placed at the front of each line. The beginning of line cannot actually go away. Using **"F1"** to mark a range of text, you can use this technique to shift code right by a fixed number of spaces when the replace string is blanks.

/[^–a–zA–Z._@$0–9\\/\\\~]/

Part of the click on a file key definition (m3), this regular express searches forward for the first character which is not part of a word. Broken down:

Character	Definition
/	Do a forward search.
[	Match on any of the following characters.
^	Invert the match, match on any but the following characters.
–	Match on a dash, must be first to avoid being taken as part of a range of characters.
a-z	Match any lower case alphabetic.

Character	Definition
`A-Z`	Match any upper case alphabetic.
`._@$`	Match any of these literal characters.
`0-9`	Match any numeric.
`\\/`	Match forward slash, extra escape characters are due to this being in a key definition. The keydef parsing eats one backslash.
`\\\`	Match a backward slash, again the extra escape characters are due to this being a key definition.
~	Match the tilde character.
]	End of the list of characters to not match.

/^.\{9,19\}abc/

Find the string "abc" in positions 9 through 19. The "^" locks the pattern to the beginning of the line. The ".\{9,19\}" matches any character "." 9 to 19 times. The "abc" matches the string abc.

s/\(.*\)/   case "\1"/

Take the rest of the line and put it in a "case" statement. The "`\(`" and "`\)`" delimit the area to assign to the internal "`\1`" variable. The "`\1`" in the replacement string shows where to put the saved part of the search expression.

Copy/Cut/Paste Operations

Chapter 1, Basics includes a section on the basic use of copy, cut, and paste operations. This section presents additional detail on the following:

- The default paste buffer.
- Named paste buffers.
- File paste buffers.
- The special "X" named paste buffer.
- A paste buffer summary.
- A special case of rectangular cutting and copying.

These sections assume that you know how to define a text region using the "Mark" keys (F1 and SHIFT/F1) and the cursor.

The Default Paste Buffer

Chapter 1, Basics described how ***Ce*** needs a default paste buffer to store information after it is copied or cut. That default paste buffer takes its name from the Ce resource dfltPasteBuf X Resource or the Ce default of "CLIPBOARD". If you exit the Ce session, the data is stored in a file in your paste buffer directory (Ce.pasteBuffDir). The file contains the data that was in the paste buffer of the Ce session in which owned the paste buffer when it was shutdown. This is how a Ce session can paste data copied in a Ce window which was closed.

When a copy operation occurs, Ce contacts the X server and requests ownership of a named entity called a selection. It remembers internally what was copied and when a paste occurs, the paster requests that the contents of the named selection be attached to its window where it can be accessed. The X server forwards the request to the selection owner who then attaches the data to the paster's window. This is standard X copy/paste, processing.

The primary limitation of the default paste buffer is that it can contain data from only a single copy or cut operation. A copy or cut to the default paste buffer always replaces any existing data in the buffer.

Named Paste Buffers

Named paste buffers behave in the same way as the default paste buffer mentioned above. That is, such paste buffers

- contain the data that was most recently copied or cut to that specific paste buffer
- are maintained via operations in the x server
- are saved in a file in your paste buffer directory

The advantage of named paste buffers is that you can do multiple consecutive copies and cuts, and have the data for each of those operations saved in distinct paste buffers. Subsequent paste operations can then address each of those paste buffers individually.

File Paste Buffers

There may be times at which you would like to copy or cut data from a file for use in non X based applications. To accommodate this need, ***Ce*** provides file paste buffers, that:

- are files created at the specified pathname (not restricted to your paste buffer directory),
- contain the data that was most recently copied or cut to that specific paste buffer, and

A file paste buffer is simply an ASCII disk file, with no other special attributes that directly relate it to ***Ce***. In fact, you can paste data from any ASCII file into a ***Ce*** window if you treat that file as a file paste buffer when using the "xp" (paste) command. For example, "xp–f~/.profile" will paste your profile into the current Ce session.

Table 3-2. Paste Buffer Summary

Operation	Commands	Default Key
Copy data		
to default pb	dr ; echo ; xc	F1, F2
rectangular to default pb	dr ; echo –r ; xc –r	<Shift> F1, <Shift> F2
to named pb	dr ; echo ; xc buf	
rectangular to named pb	dr ; echo –r ; xc –r buf	

Table 3-2. Paste Buffer Summary (Cont)

Operation	Commands	Default Key
to file pb	dr ; echo ; xc –f file	
rectangular to file pb	dr ; echo –r ; xc –r –f file	
Cut data		
to default pb	dr ; echo ; xd	F1, F3
rectangular to default pb	dr ; echo –r ; xd –r	<Shift> F1, < Shift> F3
to named pb	dr ; echo ; xd buf	
rectangular to named pb	dr ; echo –r ; xd –r buf	
to file pb	dr ; echo ; xd –f file	
rectangular to file pb	dr ; echo –r ; xd –r –f file	
Paste data		
from default pb	xp	F4
rectangular from default pb	xp –r	<Shift> F4
from named pb	xp buf	
rectangular from named pb	xp –r buf	
from file pb	xp –f file	
rectangular from file pb	xp –r –f file	

A Special Case of Rectangular Copying and Cutting

There exists one special case in which a “rectangular region” is not actually rectangular. When you initiate a rectangular region highlight via “<Shift> F1” (“dr ; echo –r”) and move the cursor to the end of the region, the highlighted area will be rectangular unless the cursor is moved to a character that is in the same column as the initially marked point. In that special case, the highlighted area includes all text in and to the right of the column defined by the initial and current cursor positions. Furthermore, lines cut or copied from such a “rectangular region” are not padded with blanks in the paste buffer to make the lines of equal length. Since the marked rectangle ends at the left edge of the text cursor, this special case corresponds to what would be a zero width copy/cut; thus no functionality is lost by the syntax.

Command Scripts

Ce provides a way to execute a file that contains one or more ***Ce*** commands. Such a construct is useful for repetitive command sequences that are too complex to assign directly to keys or too lengthy to enter each time they are to be executed.

You execute such files via the “cmdf” command, whose syntax is “cmdf[–p]file ”. “file” is an ASCII file containing a series of ***Ce*** commands. If you want to execute the commands that reside in a paste buffer, you must use the “–p” option to tell ***Ce*** to look in a paste buffer instead of a normal file.

Escape to Shell

Ce provides the ability to execute a shell or a program, and to provide to that program, text from the file you’re editing as stdin. Stdout from the shell can be automatically or manually pasted into the edit session or interpreted as a command script. See the online help on the topic “bang” for the syntactic and semantic details.

File Saving and Backup Modes

There are a number of commands, environment options, and default key definitions that are related to

- Saving a file to disk
- Closing an edit window
- Toggling a file between read–only and edit modes
- Setting the backup mode and creating a backup file
- Renaming an edit file during an edit session

Each of these operations is described in the following paragraphs. This section concludes with a table summarizing the above operations.

Saving a File to Disk

Press the “F8” key, which executes the command sequence “pw ; wc”. This sequence is actually two commands. The first, “pw”, is what causes the file to be updated to disk. The second, “wc”, is what causes the window to close. You will normally use the “F8” key to terminate an edit session.

Press “<Shift> F8”, which executes the command sequence “pw ; ro”. This sequence, also consisting of two commands, causes the file to be updated to disk (“pw”) and the window to be put into read–only mode (“ro”). This is useful when you are finished editing a file, but would like a window to remain open on it for subsequent browsing. When the window is in read–only mode, an “R” inside a small box will be present in the window’s title bar.

Press “<Ctrl> w”, which executes the command “pw” by itself. Doing this will cause the file to be updated to disk, but the edit window will remain open and receptive to subsequent edit commands. It is wise to execute this operation periodically during an edit session to minimize the risk of data loss due to a system or power failure.

Set your autosave option (either via the “–autosave” command line option or the “Ce.autosave” X resource). The option takes as an argument a positive integer n; when active, ***Ce*** will automatically update the edit file to disk (effectively executing the “pw” command) every n keystrokes.

Closing an Edit Window

If you want to close an edit window and save the changes made to the file, use the “F8” key, which is assigned the command sequence “pw ; wc”.

If you want to close an edit window without saving changes that have been made to the file since it was last written to disk, press “<Ctrl> F8”, which executes the “wc –q” command (“window close”). ***Ce*** will ask you if you really want to abort the changes. You must respond affirmatively to really terminate the edit session without saving the changes.

If the window is in read–only mode, the “pw” command, which is what causes the file to be updated to disk, has no effect. Therefore, to close a read–only window, you can press “F8” or “<Ctrl> F8”. The key sequence “<Ctrl> y” and “<Ctrl> n” are equivalent to “F8” and “<Ctrl> F8” respectively.

Toggling between Read–only and Edit Modes

Press “<Shift> F8” to change a ***Ce*** window’s mode from edit to read–only. “<Shift> F8” is assigned the command sequence “pw ; ro”. This means that before changing to read–only mode (“ro”), the file will be updated to disk (“pw”).

Press “<Ctrl> m” to change a ***Ce*** window’s mode from read–only to edit. The “R” in the window’s upper border will disappear, indicating that the window is now in edit mode.

You can also use “<Ctrl> m” to change from edit mode to read–only mode if no changes have been made to the file.

Setting the Backup Mode and Creating a Backup File

Since there may be times at which you will want to revert to a previous version of a file, ***Ce*** can automatically create a backup file before saving a changed version.

As mentioned in the Section, *Saving a File to Disk*, there are a number of events that can take place that cause a file to be updated to disk. In fact, such events can occur multiple times during a single edit session. A backup file is created when such an event occurs for the first time during an edit session.

A backup file is given the same name as the original file, except that the suffix ".bak" is added to the backup file's name. If ***Ce*** attempts to create a ".bak" file and a file with that name already exists, the existing version is overwritten. (This should not be a problem unless files with names ending in ".bak" are being generated by some other application.)

Ce will always create a backup file unless you use the "–bkuptype none" command line option or the "Ce.bkuptype: none" X resource. Backup files can be created using one of two modes:

- "dm" mode. In this mode, the original *file* being edited is renamed to "file.bak", and a new file is created. This means that any hard links that pointed to the original file will subsequently point to "file.bak". This is because a hard link actually points to a file's internal system identifier, and a file that is renamed retains its original identifier.
- "vi" mode. In this mode, file is copied to "file.bak". This means that hard links that pointed to file will still point to file, which will be the changed file following termination of the edit session. This is slower than doing the rename.

The default backup mode is "dm". Use the "–bkuptype" command line option or the "Ce.bkuptype" X resource to change the mode to "vi" or "none".

Renaming an Edit File During an Edit Session

Ce allows you to rename a file even while it is being edited. This is accomplished via the "pn" command, whose syntax is "pn [–c] pathname". Specifically, "pn" can be used in one of two ways:

- "pn path" will effectively cause the file being edited to be moved from its current path to the specified path. Moving the file in such a way means that the original path is no longer valid.
- "pn path –c" actually copies the edit file to the new path. The original file will remain at the original path, unchanged.

One special feature of ***Ce*** is that it allows you to edit "standard input". For example, you can pipe output from one command to ***Ce***'s input, e.g. "% ls –al | ce". This is a special use of ***Ce*** in which the file being edited has no associated disk file. In order to save such a file, you must use the "pn" command to assign a pathname to the file being edited.

Table 3-3. Summary of File Saving and Backup Mode Operations

Operation	Command or Option
Update a file to disk and close it	F8 ("pw ; wc")
Update a file to disk and put it in read–only mode	<Shift> F8 ("pw ; ro")
Update a file to disk and leave it in edit mode	<Ctrl> w ("pw")
Abort an edit session	<Ctrl> F8 ("wc")
Changing a window's mode from read–only to edit	<Ctrl> m ("ro")
Changing a window's mode from edit to read–only	<Ctrl> m ("ro")

Table 3-3. Summary of File Saving and Backup Mode Operations (Cont)

Operation	Command or Option
Enabling autosave mode	"–autosave n" or "Ce.autosave: n"
Enabling "dm" backup mode	"–bkuptype dm" or "Ce.bkuptype: dm"
Enabling "vi" backup mode	"–bkuptype vi" or "Ce.bkuptype: vi"
Disabling backup mode	"–bkuptype none" or "Ce.bkuptype: none"

Tab and Form Feed Character Processing

For the most part, ***Ce*** allows the editing of files that contain only ASCII characters. The sensible exceptions are the newline character (hex 0A), the tab character (hex 09), and the form feed character (hex 0C). In truth, Ce allows all character values except x' 00'. Adding key definitions utilizing the "er" (enter raw) command allow you to type these characters.

Newline characters are processed in one and only one way: they determine where line breaks occur in the text. Newline characters only occur at the end of text lines; the last line of a file is the only line that might not contain a terminating newline character.

Unlike newline characters, tab and form feed characters require special processing. Each of these special characters is described below.

Tab Characters

Within a file, a tab character is represented as a single character. When such a character is displayed in the edit window, however, it is expanded to appear as a string of blanks so that text following the tab character starts at the next tab stop.

The expansion is determined solely by the tab stop settings. If your file contains tab characters and you change the tab stop settings during an edit session, lines containing tab characters will be automatically adjusted to the new tab settings.

There are three default key definitions related to tab processing:

- "Tab", defined as "th". Pressing this key causes the cursor to move to the next tab stop to the right. The text is not modified.
- "<Shift> Tab", defined as "thl". Pressing this key causes the cursor to move the next tab stop to the left. The text is not modified.
- "<Ctrl> Tab", defined as "er 09", which inserts a tab character at the current cursor position.

There may be times at which you do not want tabs to be expanded as explained above, but would rather have tabs displayed as themselves. Use the “hex” Ce command to disable tab character processing and view unprintable characters. (As noted below, if tab processing is disabled in this way, form feed character processing is also disabled.)

Form Feed Characters

If tab processing is enabled (the default), form feed character processing is also enabled. A form feed forces the line on which it occurs to be displayed as the first line in the window, having the effect of starting a new page.

If a file contains multiple form feed characters, only one such character can occupy the first line and column of the window at one time. The only way in which the current form feed character will give up its position is if it is scrolled out of the window. When that happens, if text that is scrolled into the window contains a form feed character, the line that contains it is forced to the top of the window.

The extreme example of this is a file in which every line contains a form feed character, for such a file can be viewed only a single line at a time.

Editing Non–ASCII Characters

As mentioned before, Ce can process all characters except x’00’. Most keyboards, however, have no way of typing non-ASCII characters. Some keyboards use control sequences to enter non-ASCII characters such as the special characters used in European alphabets. The ISO latin option is used to prevent these sequences from being filtered out. Other non-ASCII characters are just not on the keyboard. To type these characters requires setting up your own key definitions.

For example, suppose you want to be able to type the character x’02’ and want to use the key sequence “<Ctrl> 2” to enter it. You would execute the key definition “kd ^2 er 02 ke”.

Character	Definition
kd	The Ce key definition command
^2	Key specification for the key 2 with the <Ctrl> modifier
er	The Ce enter raw command to be assigned to this key sequence
02	The parameter to er, the hex value to enter
ke	End of the key definition marker

Editing the same document on multiple workstations

Often times it is convenient for more than one person to edit a file at one time. Having a friend proofread code or a memo without printing out hard copy, having it marked up, and then entering the changes cuts out several steps in the process. It is especially convenient if the other person is physically close so you can talk while you type.

ARPUS/Ce provides the “cc” command to facilitate this type of operation. The “cc” command takes most of the arguments available to ce and cv. This includes the “–display” option. Without the display option, the “cc” command creates an additional window on your workstation for the current file being edited. This allows two or more views into the file. For example, it may be nice to look at data declarations at the top of a file at the same time as working on code near the bottom. ***Ce*** theoretically supports any number of concurrent windows, but you will run out of real estate on the screen after a few.

If the “–display node:0” option is used on a “cc” command, the view into the file will be opened on the workstation specified by “node”. The user of that workstation must allow the current workstation to open windows on its display with the UNIX “xhost” command. If the the other nodes display has not been so unlocked, the results vary. On most systems, an error message specifying that the connection has been rejected will will be generated. On some systems, the window will hang for several minutes until a timeout message is generated. This is a function of the X server implementation on the target node.

Once a window has been opened on the other display, the other user is in edit on the same memory image of the file. Either user can save the file, make changes, etc. If both users are viewing the same lines in the file they can see what the other user is typing. (This is amusing the first time you see it.) One restriction is that both users cannot type on the same line at the same time. Each user uses the key definitions and X defaults associated with the node they are on. Thus one user could be using Aegis regular expressions and one standard UNIX regular expressions. This functionality can extend to multiple workstations. In theory any number is possible, but it has not been tried with more than 10 workstations.

The “cc” command also works with a ceterm. In this case only the transcript pad is displayed in the cc window along with a ***Ce*** “Command:” window and an output message window. This can be useful when providing support to a ***Ce*** user who is having some problem and has called on the phone. Open the current workstation to allow windows to be opened from the other workstation and have them do a “cc –display your node:0” and a copy of their transcript pad will be placed on your workstation. You can then see the commands as they execute them. This really helps when the user types “ls–la” instead of “ls –la”.

WARNING: When you open a window on another node, the other user is editing the file as you on your workstation . They can execute the “ce”, “cv”, “!”, “ceterm”, “cpo”, etc. commands and create other processes running as you on your workstation. In some environments, this is not an issue. In others it is an issue. Use due caution. If you are running as root, you should probably not open windows on other peoples displays.

Record / Playback Modes – Learn Mode

ARPUS/Ce supports recording keystrokes (and mouse movement) and later playing them back as a command file. The actual keystrokes are not recorded, but instead the ***Ce*** commands stored under them are played back. Some editors refer to this as learn mode. While the default pulldown menus allow access to record and playback, most users choose to put them under key definitions.

For this example, let us assume that you have executed the UNIX command, “xmodmap –e ”remove mod3 = Num_Lock” in your “.vueprofile (HP/VUE)”, “.dtprofile (CDE”), or “.xinitrc” (others) file. This disables the shift-like modifier function of this key and frees it up for use as a normal key. We will use the “Num_Lock” key to turn on record mode, the key next to it (key pad divide) to turn it off, and the next key (key pad multiply) to do a playback. Many people find this arrangement convenient, but it is impractical to ship ***Ce*** with this as the default.

We now place the following key definitions into the .Cekeys file and run ’ce –reload’ from the UNIX command line.

kd Num_Lock	rec –p cemacro ke	# macro record on
kd KP_Divide	rec ke	# macro record off
kd KP_Multiply	cmdf –p cemacro ke	# macro play

Suppose we want to insert a “/*” and three blanks at the beginning of each of a selected set of lines and three blanks and a “*/” at the end.

1. Position the cursor at the beginning of the first line to operate on.
2. Press Num_Lock to turn on recording.
3. Press Home to go to the beginning of the line.
4. Type “/*   “.
5. Press the ’End’ key to go to the end of the line
6. Type “   */”.
7. Press the ’Home’ key to go back to the beginning of the line.
8. Press arrow down to go to the next line. Note it may be useful to have arrow down defined as *kd Down [+1,] ke*, so the window will scroll when you get to the bottom of the page.
9. Press the key pad divide (/) key to turn off recording.
10. Press key pad multiply (*) to repeat the operation on the next line.

The recorded command sequence will work in any ***Ce*** window until it is rerecorded over. You can see the recorded command sequence by pasting the paste buffer used by the record key. You would execute the command “xp cemacro” to perform the paste. You can also record to a real file. See the help on “rec” for more information.

Chapter 4: The “ceterm” Window

Organization of This Chapter

This chapter explains the ceterm window, which is a visual tool that you use to interact with a command shell such as the Korn shell.

This chapter describes the following:

- Starting a ceterm window
- ceterm window description

Starting a ceterm Window

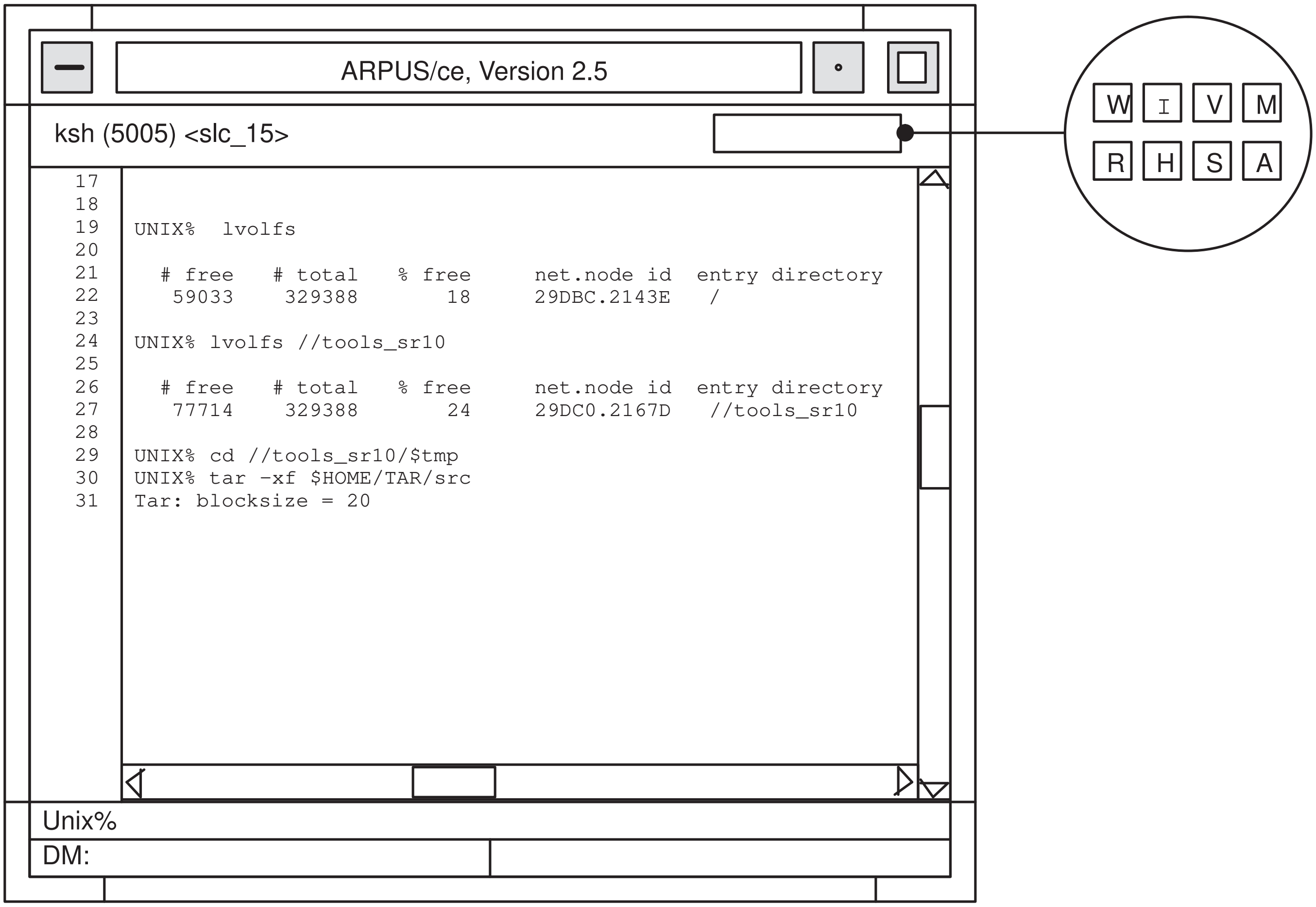
As shown in Figure 4-1, a ceterm window closely resembles a ***Ce*** window and includes several additional components. The ceterm window is used in very much the same way as the various vendors’ visual shell tools (e.g., DEC’s decterm, HP’s hpterm, IBM’s aixterm, Sun’s shell tool).

You can start a ceterm window in any of the following ways:

- Type “ceterm” at any shell prompt.
- Type “ceterm” at the “Command:” prompt of any ***Ce*** window.

The primary addition is the shell input window, which is the window into which shell commands are typed. The shell prompt appears in this input window. This window is actually a single-line ***Ce*** edit window that accepts the same edit operations that you can perform while editing a file.

Output from the shell and from the commands executed within the shell appears in the transcript pad. The transcript pad is simply a read-only ***Ce*** edit window. That is, all of the operations that you can perform on a read-only ***Ce*** window (copying, scrolling, cursor movement, etc.) can be performed within the transcript pad.



W – Word wrap mode (ce only)

I – Insert mode

M – File has been modified (ce only)

V – VT100 mode (ceterm only)

R – File is read only (ce only)

H – Hold mode (ceterm only)

S – Line scrolling mode turned on (ceterm only)

A – Auto scrolling turned on (ceterm only)

NOTE: The scroll bars can be toggled on and off with the “sb” command and the line number column can be toggled on and off with the “lineno” command. The “lineno” and “scrollBar” X resources can be used to set the initial state of line numbers and the scroll bars.

Figure 4-1. Appearance of ceterm Session

ceterm Window Description

As you can see in Figure 4-1, a ceterm window is based on the ***Ce*** window but includes several additional components related strictly to ceterm. In *Chapter 1, Figure 1-1,* the main window (component number 9) was called the edit window. In a ceterm window the main window that receives the command output is referred to as the transcript pad. The ceterm window also includes the following:

- The "H"character in the title bar, which is the hold indicator. The presence of the "H" indicates that the window is currently in hold mode, which means new output from the shell commands is held until either you explicitly scroll the transcript pad, or you take the window out of hold mode (via the "wh" command or the "<Ctrl> h" key).
- The "S" character in the title bar, which is the scroll mode indicator. The presence of the "S" indicates that the transcript pad is scrolled one line at a time as each line is received from commands or applications running in the ceterm window. The absence of the "S" means that the transcript pad is scrolled one window at a time.
- The "A" character in the title bar, which is the autohold indicator. The presence of the "A" indicates that the window is automatically put in hold mode whenever the transcript pad receives a single window's worth of output. The window remains in hold mode until you take it out of hold mode via the "wh" command or the "<Ctrl> h" key.

In addition to the new visual cues accompanying a ceterm window, ARPUS/ce includes several commands related specifically to ceterm windows but not to ***Ce*** windows. Each of the commands described below have corresponding on–line help systems which can be viewed by typing pressing "F9" (or the "Help" key on Sun keyboards) and typing the command name at the "Help on:" prompt.

cp, cpo, cps

These commands create processes and accept command names and arguments in the same form that you would use in a shell. "cp" is synonymous with ceterm, which creates a new process and a corresponding ceterm window within which to run the process (usually a command shell such as "/bin/ksh"). "cpo" and "cps" both create background processes that have no windows associated with them.

eef

This tells ceterm to insert and end–of–file character, usually used to terminate a program's standard output or, more specifically, to terminate the current ceterm and corresponding shell. The "<Ctrl> d" key is defined to insert an end–of–file character, just as "<Ctrl> d" is used in a normal shell window.

env

This command takes as an argument a shell environment variable and displays in the ceterm editor output window the value of that environment variable, if any.

vt

This command puts the ceterm window in vt mode to simulate a virtual terminal. Programs that use curses and other vt-style control methods can be run while in vt mode. While in vt mode keystrokes are passed by ceterm directly to the application; even the "Escape" key, normally used to move the cursor to the "Command:" prompt, is passed on. Therefore to get back out of vt mode into ceterm mode, the cursor must be moved via the mouse to the "Command:" prompt, at which point the "vt" command can be executed again.

wa

This command puts the ceterm window in autohold mode, in which the window is automatically put in hold mode whenever a windowfull of output is received in the transcript pad. This command is normally invoked via the “<Ctrl> a” keystroke.

wh

This command puts the window into or out of hold mode. It is normally invoked via the “<Ctrl> h” key.

ws

This command puts the window into or out of scroll–by–line mode. It is normally invoked via the “<Ctrl> s” key.

As evidenced by the constant reference to key definitions, one of the best ways to learn the behavior of these commands is to learn which keys they are associated with and use those keys in a practice session. You can easily determine key definitions by referring to *Appendix A, Key Summary*, which lists the key definitions for a default installation of ***Ce*** and read your personal “$HOME/.Cekeys” file, which was automatically installed for you when you ran “ce_init” as explained in *Chapter 2, Environment Customization*.

Appendix A: Key Summary

NOTE: The key names shown below do not necessarily correspond to the X names for those keys on your system. The “ce_init” shell command installs a “.Cekeys” file in your home directory. That file contains key definition commands corresponding to the definitions shown below.

Table A-1. Function Keys

Key	Definition
F1	Mark the start of a text region
F2	Copy a text region into the default paste buffer
F3	Cut a text region into the default paste buffer
F4	Paste text from the default paste buffer
F5	Delete the current line into the “line_del” paste buffer
F6	Paste the contents of the “line_del” paste buffer
F7	Undo
F8	Write the file to disk and close the window
F9	Prompt for a Help topic and create a read-only window on it.
<Shift> F1	Mark the start of a rectangular text region
<Shift> F2	Copy a rectangular text region into the default
<Shift> F3	Cut a rectangular text region into the default
<Shift> F4	Paste a rectangular block of text from the default
<Shift> F5	Delete to the end of line into the “eol_del” paste buffer
<Shift> F6	Paste the contents of the “eol_del” paste buffer
<Shift> F7	Redo
<Shift> F8	Write the file to disk and put change to read–only
<Ctrl> F1	Restore last highlight
<Ctrl> F5	Delete the word under the cursor to paste buffer “word_del”
<Ctrl> F6	Paste the contents of the “word_del” paste buffer
<Ctrl> F8	Abort the edit session

Table A-2. Numeric Pad (NP) Keys

Key	Definition
NP1	Move the cursor to the beginning of line
NP2	Move the cursor down
NP3	Move the cursor to the end of line
NP4	Move the cursor left
NP5	Move the cursor to the beginning of line
NP6	Move the cursor right
NP8	Move the cursor up
<Shift> NP2	Scroll the window down 1 line
<Shift> NP4	Scroll the window left 1 column
<Shift> NP5	Move the cursor to the end of line
<Shift> NP6	Scroll the window right 1 column
<Shift> NP8	Scroll the window up 1 line
<Ctrl> NP2	Scroll the window down a half page
<Ctrl> NP4	Scroll the window left 10 columns
<Ctrl> NP6	Scroll the window right 10 columns
<Ctrl> NP8	Scroll the window up a half page

Table A-3. Specially Labeled Keys

Key	Definition
Esc (Escape)	Move the cursor to the “Command:” prompt
Return	Insert a newline character at the current cursor position
<Shift> Return	Insert a blank line below the current line
Tab	Move the cursor to the next tab stop to the right
<Shift> Tab	Move the cursor to the next tab stop to the left
<Ctrl> Tab	Insert a tab character at the current cursor position
Backspace	Delete the character to the left of the cursor
Delete	Delete the character under the cursor

Table A-3. Specially Labeled Keys (Cont)

Key	Definition
Help	Prompt for a Help topic and create a read-only window on it.
Insert	Toggle insert / overstrike mode
Home	Move cursor to start of current line
End	Move cursor to end of current line

Table A-4. Control/Alpha Keys

Key	Definition
<Ctrl> a	Toggle autohold mode
<Ctrl> b	Move the cursor to the bottom of the file
<Ctrl> c	Send quit–fault
<Ctrl> d	Insert end–of–file character
<Ctrl> e	Repeat substitute once
<Ctrl> h	Toggle hold mode
<Ctrl> i	Toggle the window between insert and overstrike modes
<Ctrl> k	Move the cursor to the beginning of the next line
<Ctrl> m	Toggle the window between edit and read–only modes
<Ctrl> n	Abort the edit session
<Ctrl> p	Print a marked range of text
<Ctrl> q	Send quit–fault (Apollo version of ^c)
<Ctrl> r	Search forward for the most recent search pattern
<Ctrl> s	Toggle scroll–by–line mode
<Ctrl> t	Move the cursor to the top of the file
<Ctrl> u	Search backward for the most recent search pattern
<Ctrl> v	Copy a text region into the default paste buffer
<Ctrl> w	Write the file to disk
<Ctrl> x	Abort the current search or highlight operation
<Ctrl> y	Write the file to disk and close the window

Table A-5. Mouse Buttons

Button	Definition
Left	Mark the start of a text region, Copy on button release
Middle	Paste text from the default paste buffer
Right	Create read-only window on the file whose name is currently pointed at by the cursor
<Shift> Left	Mark the start of a rectangular text region, Copy on release
<Shift> Middle	Paste a rectangular block of text from the default
<Shift> Right	Create editable window on the file whose name is currently pointed at by the cursor
<Alt> Middle	Copy a word to the default paste buffer
<Ctrl> Left	Do a find on the contents of the default paste buffer and save the find command
<Ctrl> Middle	Repeat the find command saved in <Ctrl> Left
<Ctrl> Right	Cut a word and replace it by the contents of the default paste buffer

Table A-6. Alt/Alpha Keys

Key	Definition
<Alt> b	Find balancing paren, brace, bracket, etc.
<Alt> c	Copy a text region into the default paste buffer
<Alt> e	Cut a text region into the default paste buffer
<Alt> p	Paste text from the default paste buffer
<Alt> r	Search forward for the next colored area
<Alt> t	Text Flow a rectangularly marked region of text
<Alt> u	Search backward for the previous colored area

Table A-7. Sun Type 5 Keyboard

Key	Definition
Help	Prompt for a Help topic and create a read-only window
Again	Copy from cursor pos to end of line to UNIX input window
Props	Show name of next key pressed
<Shift> Props	Show name of next 2 keys pressed
Undo	Undo last action
<Shift> Undo	Redo last action
Front	(Window Manager controlled key – not available to ***Ce***)
Open	(Window Manager controlled key – not available to ***Ce***)
Copy	Copy marked region into the default paste buffer
<Shift> Copy	Rectangular copy of marked region into the default paste buffer
<Ctrl> Copy	Copy marked region into the alternate paste buffer
<Ctrl> <Shift> Copy	Rectangular copy of marked region into the alternate paste buff
Paste	Paste the default paste buffer
<Shift> Paste	Rectangular Paste of the default paste buffer
<Ctrl> Paste	Paste the alternate paste buffer
<Ctrl> <Shift> Paste	Rectangular Paste of the alternate paste buffer
Cut	Cut marked region into the default paste buffer
<Shift> Cut	Rectangular cut of marked region into the default paste buffer
<Ctrl> Cut	Cut marked region into the alternate paste buffer
<Ctrl> <Shift> Cut	Rectangular cut of marked region into the alternate paste buff
Find	Search forward for the most recent search pattern (repeat find)
<Shift> Find	Search backward for the most recent search pattern

This page intentionally left blank

Appendix B: Command Summary

The commands in the table below have corresponding on–line help files. Those on–line help files contain fuller descriptions, since it is likely that you will need such descriptions during your actual edit sessions. Most help files are named after their respective commands. For cases in which that is not possible (e.g. “//”) or practical (e.g. “(x,y)”), the table entry shows the name of the on–line help file.

The keys noted in this table indicate keys that are related to the corresponding commands; it is not necessarily the case that those keys perform exactly those commands. For example, the table shows that “F8” is related to the “wc” command, and F8’s definition is “pw ; wc”.

Table B-1. Command Summary

Command	Description	Accepts Arguments	Works with marked range of text	Related Keys
! command	Execute command in a shell *(Help file: bang)*	√	√	<Ctrl> p (print)
$ command	Go to bottom of file *(Help file: dollar)*			
(x,y)	Position cursor at screen pixel coordinates (x,y) *(Help file: pixelpos)*	√		
<n>	Move cursor to specified line *(Help file: line)*	√		
=	Show coordinates of the current cursor position *(Help file: pos)*			
[row,col]	Position cursor at [row,col] in window *(Help file: rowcol)*	√		
{row,col}	Position top left of window at [row,col] *(Help file: corner)* *(Help file: rfind)*	√		
/pattern/	Search forward for pattern *(Help file: find)*	√		
//	Search forward for the previous search pattern *(Help file: find)*			<Ctrl> r
\pattern\	Search backward for pattern (AEGIS syntax) *(Help file: rfind)*	√		
?pattern?	Search backward for pattern (UNIX syntax) *(Help file: rfind)*	√		

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu-ments	Works with marked range of text	Related Keys
\\	Search backward for previous search pattern *(Help file: rfind)*			<Ctrl> u
??	Search backward for previous search pattern (UNIX syntax)			<Ctrl> u
abrt	Abort current search or highlight operation			<Ctrl> x
ad	Move cursor down 1 line			Arrow Down
al	Move cursor left 1 character	√		Arrow Left
alias	Alias commands	√		
ar	Move cursor right 1 character	√		Arrow Right
au	Move cursor up 1 line			Arrow Up
bell	Beep the bell			
bgc	Set background color			
bl	Balance delimiters in a text region	√		
ca	Color area of text	√	√	
caps	Enable caps-only mode	√		
case	Switch the case of letters in a text region	√	√	
cc	Carbon copy	√		
cd	Change editor directory			
ce	Create an edit window on a file	√		Right Mouse Button
ceterm	Create a new ceterm window	√		
cmdf	Execute the commands in a file	√		
cms	Clear the mark stack			
cp	Create a new ceterm window	√		

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu-ments	Works with marked range of text	Related Keys
cpo	Create a new process without a win-dow	√		
cps	Create a new process without a win-dow	√		
cv	Create a read–only window on a file	√		
dlb	Delete labels created by lbl in an area		√	
dq	Send a quit or kill signal to a process	√		<Ctrl> q, <Ctrl> c
dr	Mark the current cursor position			F1, <Shift> F1
echo	Highlight text between a mark and the cursor	√		
ed	Delete the character under the cursor		√ (Depends on settings)	Delete
ei	Put the window into insert or overstrike mode	√		Insert
ee	Delete the character to the left of the cursor	√	√ (Depends on settings)	Back-space
eef	Insert an end–of–file character			<Ctrl> d
en	Insert a newline at the current cursor position			Return, Enter
env	Display/set an environment variable	√		
envar	Toggle substitution of environment vars in cv and cmdf	√		
er	Insert the character whose hex code is given	√		
es	Insert a text string at the current cursor position	√		

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu- ments	Works with marked range of text	Related Keys
eval	Substitute environment vars and run Ce commands	√		
f	Forward search	√		
fbdr	Adjust find border value	√		
fgc	Set foreground color	√		
fl	Load a font	√		
geo	Set/get the window’s size and location			
glb	Go to named label created with lbl	√		
gm	Go to the most recently marked point			
hex	Enable or disable hex mode	√		
icon	Iconify the current window	√		
inv	Swap the foreground and background colors			
kd	Assign/inspect a key definition to/for a key	√		
keys	Display current key definitions			
kk	Show the name of the next key that is pressed			(Sun Props)
lbl	Assign a label to a location in a file	√		
lineno	Show or hide the line number column	√		<Ctrl> l
lkd	Key definition local to one window	√		

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu-ments	Works with marked range of text	Related Keys
lmi	Menu Item local to one window	√		
mi	Add an item (line) to a pulldown menu	√		
mouse	Disconnect mouse cursor from text cursor	√		
msg	Display a message in the command output window	√		
nc	Move the cursor to the next colored text	√		<Alt> r
pb	Scroll the last line of the file into the window			<Ctrl> b
pc	Move the cursor to the previous colored text	√		<Alt> u
pd	Activate a pull down menu	√		
pdm	Toggle displaying the Pull Down Menu bar	√		
ph	Scroll the window horizontally	√		<Shift> Left arrow, <Shift> Right arrow
pn	Move or copy the edit file to a new pathname	√		
pp	Scroll the window by pages	√		Page Up Page Down
prefix	Set scan prefix	√		
pt	Scroll the first line of the file into the window			<Ctrl> t
pv	Scroll the window vertically by lines	√		<Shift> up arrow <Shift> down arrow

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu-ments	Works with marked range of text	Related Keys
pw	Write the file to disk			F8, <Shift> F8
pwd	Display current Ce working directory			
rd	Redraw window immediately			
re	Translate regular expression from Aegis to UNIX	√		
rec	Record commands	√		
redo	Redo a change that was undone via “undo”			<Shift> F7
rm	Restore the most recent mark to the stack			<Ctrl> F1
ro	Toggle the window between edit and read-only	√		<Shift> F8, <Ctrl> m
rs	Refresh window contents of all visible windows			
rw	Refresh the window contents			
s/p1/p2/	Replace p1 with p2 throughout a text region	√	√	
sb	Turn scroll bars on and off	√		
sc	Set case sensitivity on search	√		
sic	Move the text cursor the mouse cursor position	√		
sleep	Delay processing	√		
so/p1/p2/	Replace the first occurrence of p1 with p2 on each line	√	√	
sp	Set UNIX window prompt – for “Ce aware” programs	√		
sq	Quit search in progress			
tb	Move cursor to bottom of window			

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu-ments	Works with marked range of text	Related Keys
tdm	Move cursor to the “Command:” prompt			Escape
tdmo	Move the cursor to the editor output window			
tf	Justify text region	√	√	
th	Move cursor to next tab stop to the right			Tab
thl	Move cursor to next tab stop to the left			<Shift> Tab
ti	Move the cursor to the main input window			
title	Change the Window Manager title	√		
tl	Move cursor to the beginning of line			Home
tmb	Move the cursor to the menu bar and freeze			
tmw	Move the cursor to the main window			
tn	Move the cursor to the next Ce window			
tr	Move cursor to the end of line			End
ts	Set tab stops	√		
tt	Move cursor to top of window			
twb	Move cursor to a window border	√		
undo	Undo the most recent change			F7
untab	Convert tab characters to blanks in a region of text		√	
vt	Toggle to or from vt mode	√		

Table B-1. Command Summary (Cont)

Command	Description	Accepts Argu- ments	Works with marked range of text	Related Keys
wa	Toggle to or from window autohold mode	√		<Ctrl> a
wc	Close the edit window	√		F8, <Ctrl> F8
wdc	Define a default color combination	√		
wdf	Define a default window geometry	√		
wh	Toggle to or from hold mode	√		<Ctrl> h
wp	Pop the window to the top or bottom of the stack	√		
ws	Toggle to or from scroll–by–line mode	√		<Ctrl> s
ww	Enable or disable window wrap mode	√		
xa	Append one paste buffer to another	√		
xc	Copy a text region into a paste buffer	√	√	F2, < Shift> F2
xd	Cut a text region into a paste buffer	√	√	F3, <Shift> F3
xl	Copy literal	√		
xp	Paste a text block from a paste buffer	√		F4, <Shift> F4

Appendix C: Customization

Introduction

As illustrated throughout this document and the on–line help files, keys that are pressed are actually translated into sequences of ***Ce*** commands. Furthermore, you can customize those command sequences to suit your own editing style and purposes.

When you ran “ce_init”, it created in your home directory a file called “.Cekeys”. (If you did not run “ce_init”, please refer to *Chapter 1, Editing Basics*, which describes the initialization process.)

“.Cekeys” contains two sets of key definitions:

- One set assigns definitions only to keys that are common to all keyboards (F1, F2, ..., F8, Escape, Tab, Backspace, Delete, Ctrl, Shift, and Return). The advantage of such a key definition set is that you can use ***Ce*** on several different platforms and get the same behavior from the keys.
- Another set assigns definitions to keys that are special for the keyboard you are using. For example, the Sun Type 5 keyboard contains keys that are labeled “Cut”, “Copy”, “Paste”, and “Undo”, among others. Since you indicate to “ce_init” the type of workstation you are using, it can copy those extended key definitions into your “.Cekeys“ file so that those special keys function properly while using ***Ce***.

Key Names

Each key has a name, also known as a “keysym” in X terminology. When you create your own key definitions, you will need to know the names of the keys to which you are assigning those definitions. You should be aware that key names vary from system to system, even if those names refer to the same key. For example, the traditional “return” key might be known as “Return” on one system and “Enter” on another system. ***Ce*** provides the “kk” command to find the names of keys on the system you are using. You can also execute the command “xmodmap –pk” from a shell to get a listing of the keysyms currently recognized by the X server.

Key Modifiers

A pressed key can be modified by simultaneously pressing Shift, Ctrl, Mod1, Mod2, Mod3, or Mod4 keys. Modified key presses are treated distinctly from unmodified key presses. For example, ***Ce*** will treat “<Shift> F8” differently than “F8”. You should also realize that modifier keys can be combined, so that “<Shift><Ctrl> F8” is another distinct key press. The epitome of such key presses is when you press all six modifier keys along with a non–modifier key, e.g., “Mod1/Mod2/Mod3/Mod4<Shift><Ctrl> F8”.

Since the X environment is very flexible, you can define any key on your keyboard to mean “Shift” or “Ctrl”. However, “Shift” and “Ctrl” are universally assigned to the physical keys with those labels. The remaining four modifier keys (Mod1, Mod2, Mod3, Mod4) are not assigned to keys in a way that is consistent across platforms. You should run the command “/usr/bin/X11/xmodmap” to learn which keys on your keyboard serve as the Mod1, Mod2, Mod3, and Mod4 keys.

Modified Key Names

The key name is required when assigning a ***Ce*** command sequence to a key. If you want to assign a command sequence to a modified key, you need to add prefixes and/or suffixes to the name to arrive at a modified name. For example, if the key name for the function key F8 is “F8”, the correct name for the F8 key modified by Shift is “F8S”. Table 8.1 lists the prefixes and suffixes used to modify key names:

Table C-1. Key Name Prefixes and Suffixes

Modifier name	Modifier symbol	Prefix or Suffix
Shift	S (uppercase S)	Suffix
Ctrl	^ (caret)	Prefix
Mod 1	* (asterisk)	Prefix
Mod 2	~ (tilde)	Prefix
Mod 3	= (equal sign)	Prefix
Mod 4	“ (backquote)	Prefix

Table C-2. Key Name Prefix and Suffix Examples

To specify	You would use
<Shift> Return	ReturnS
<Ctrl><Shift> F8	^F8S
Mod1/Mod2/BackSpace	*~BackSpace
<Shift><Ctrl>Mod1/Mod2/Mod3/Mod4/Tab	^*~=‘TabS

The order of appearance of the prefix modifier symbols is not important. What is important is that you press all of the modifier keys and hold them immediately prior to pressing the primary key.

The terms Mod1, Mod2, etc. can be confusing. These are the names the “X” documentation uses. In general, Mod1 is the <Alt> key. This is pretty consistent across vendors and keyboards. On Sun, the diamond key is usually Mod4 although on some older releases of SunOS, Alt is mod4 and the diamond key is Mod1. The ***Ce*** “kk” command can be used to sort this out for your keyboard. See the online KK command help and the description below for more information.

The “kd” (Key Definition) Command

You use the “kd” command to assign a command sequence to a key. Your “.Cekeys” file is simply a series of “kd” commands that are processed when you start ***Ce***. The following lines are taken directly from a typical “.Cekeys” file:

```
kd F1     dr ; echo ke
kd F1S    dr ; echo –r ke
kd F7     undo ke
kd F7S    redo ke
kd F8     pw ; wc ke
kd F8S    pw ; ro ke
kd ^F8    wc –q ke
```

The basic syntax for the “kd” command is “kd <name> <commands> ke”, where <name> is an unmodified or modified key name according to the rules in the previous section, and where <command> is a valid sequence of ***Ce*** commands, separated by semicolons.

If you want a key definition to be in effect every time you run ***Ce***, you should place the definition in your “.Cekeys” file. ***Ce*** processes the key definitions in that file before handling events from your keyboard or mouse.

You can also issue “kd” commands in the command window at the “Command: ” prompt during your edit session. Such key definitions, however, apply only to the current session and are not saved across edit sessions.

The “kk” (Key Key) Command

The Ce “kk” command is used to determine the name you would use in a key definition for a particular key or combination of keys. After a “kk” is executed the keysym name corresponding to the next key pressed is displayed in ***Ce*'s** output window. If two “kk” commands are stacked, as in “kk;kk”, the next two keystrokes are displayed. One of these may be the pressing of a modifier key.

For example: In the ***Ce*** command window type “kk;kk” and press “Return”. Press and hold the “Control” key. The ***Ce*** output message window should display something like “CONTROL_L”. Then press the “a” key. The ***Ce*** output message window should now display “^a”. This value, “^a”, is what you would use in a key definition to define the action when the sequence Control a is pressed.

The “kd” command requires a keysym to bind a definition to a specific key or key combination. By default, on a Sun keyboard, the key with the word “Props” on it executes a “kk” command and Shift Props executes “kk;kk”.

NOTES:

1.. The window manager that you use (OSF/Motif, Open Windows, ...) can reserve certain keys which are then unavailable for use by applications such as ce. If you find that one of your key definitions is not functioning as expected, consult your window manager documentation to see if you are using such a reserved key or key combination. You can also use the “kk” command to determine this. Execute a “kk” (or “kk;kk”) and then press the key or key combination in question. If you do not get the messages in the ***Ce*** output window, then ***Ce*** did not receive this keystroke. Thus the window manager is intercepting it.

2.. If the environment variable “CE_FULL_KK” is set, “kk” will display the hardware keycode and the default es string following the keysym. In the “Command:” window execute: “env CE_FULL_KK=1”.

3.. The key mapping on HP/UX keyboards uses the alt key to map to many of the multi-national keys. For example, the “<Alt> h” key combination usually maps to the Japanese Yen symbol. On an IBM RS6000, with the Alt key pressed down, you would use “*h” to define “h”, on an HP you would use “*yen”.

The alias Command

The alias command has a syntax almost identical to the key definition “kd” command. The word “alias” replaces “kd” and the name of the alias replaces the name of the key. The “alias” command allows the user to create additional ***Ce*** commands which may be used in key definitions or typed at the command line. The three forms of the “alias” command are as follows:

alias “name”	Display in ***Ce’s*** output window the definition currently assigned to the specified alias.
alias “name” “def” ke	Alias name to be “def”. “def” can be a list of commands, may contain prompts, and may contain positional parameters to be substituted in later. “def” is parsed when the alias is executed.
alias “name” ke	Erase any definition associated with name.

NOTE: “name” is a 1 to 8 character alias name composed of all alphabetic characters. It must not be the same as any built in ***Ce*** command.

“<def>” is a string of ***Ce*** commands separated by semicolons. It is parsed in exactly the same way as a “kd” command. It may contain prompts just like key definitions. In addition, it may contain positional symbolic parameters ($1, $2, etc.) which will be substituted in when the alias is invoked. That is “$1” refers to the first word following the alias command, “$2” refers to the second word, and so forth.

In addition, the constructs “$*” and “$@” are supported. The “$*” construct resolves during alias expansion to all the supplied arguments with a space between each argument. The “$@” construct resolves to all the supplied arguments with a set of double quotes (”) around each argument and a space between each argument.

Alias’ed names may be used in key definitions. However, the alias must already be defined and the alias is expanded during the processing of the key definition. Thus, aliases which use symbolic substitution need to have their parameters specified within the key definition. Examples are provided below:

alias help cv $CEHELPDIR/$1.hlp ke

This alias creates a command named help which opens a cv window on the help file named as the first and only argument to the command.

alias quit wc –q ke

This alias creates a command named quit which closes the a ***Ce*** session without saving any changes.

alias print 1,$! –c –m lp k

This alias creates a command named print which will print the entire file by piping it into the system “lp” command.

The mi (menu item) command

The “mi” command is used to set, display, or erase a menu item in the named pull down menu. Changes made via the “mi” command are propagated to all ***Ce*** windows on the current display. Note that the “lmi” (Local Menu Item) command behaves just like the “mi” command except that the changes are local to the window they are executed in. The three forms of the “mi” command are as follows:

mi <pdname>/<line>

Display in ***Ce’s*** output window the definition currently assigned to the specified menu item.

mi <pdname>/<line> ’title’<ce;cmd;list> ke

Add or replace a menu item with a new definition. “<ce;cmd;list>” is parsed at definition time for validity, and if ***Ce*** detects a syntax error in the definition, it is rejected. Using the value “$” for the line number causes ***Ce*** to find the next available line in the menu item and use that line. This includes items previously erased. If a menu item in the named pull down exists with the same title, the menu item replaces this item no matter what was specified in the line.

mi <pdname>/<line> ke

Erases a menu item. The line is not displayed but exists internally. Using “–1” as the value for line causes all the menu items in that pulldown to be erased.

Parameters

<pdname>

This is the name of the pulldown to be modified or added to. It is 1 to 8 alphanumeric characters. You can make up new pulldown names and add Menu Items to them. They can then be chained to existing pulldowns or activated from a key definition “kd”, command file “cmdf” or from the “Command:” prompt. The “pd” (Pull Down) command causes a pulldown to be activated. To add menu items to the default pulldowns, you need to know their names. From left to right, the pull down names are: File, Edit, Modes, Macro, and Help. Their names are the same as their title in root pulldown whose name is “Menubar”. Note that in the processing of the root pulldown Menubar, item “Help” is treated as a special case.

<line>

This is the zero based line number of the menu item to examine, update, or add. When you add lines to an existing pulldown, make sure you do not skip any intermediate line numbers. The lines, starting at zero, are checked until one which is not found is encountered. This is taken to be the end of the list. Thus a hole in the numbering sequence terminates the list.

<title>

This is the string to be placed into menu item line. It is what is displayed on that line in the pull down. The string can be enclosed in single, double, or reverse quotes. If a menu item in the named pull down exists with the same title, the menu item replaces this item no matter what was specified in the line.

<ce;cmd;list>

This is the list of ***Ce*** commands separated by colons to be executed under this menu item. It is parsed just like a key definition. For an arbitrary list of ***Ce*** commands, the commands are executed when a the mouse button is released in the pulldown menu item corresponding to this list. Holding the left mouse button down, and dragging to a menu item or clicking a menu item produce the same result.

The exception to the button release rule is when the ***Ce*** command list contains a single “pd” command. In this case, the pulldown command is executed when the mouse button is pressed or when the mouse button is held down and the mouse is dragged into the pulldown menu item. This exception gives the pulldowns the expected behavior. A side effect of this is that you are not allowed to mix a “pd” command in the command list with anything else. The ***Ce*** command list can either contain a single “pd” command or a list of other ***Ce*** commands. Using cmdf, aliases, eval, or other techniques it is possible to hide the “pd” command from the “mi” check which disallows this mix. The effect of this is that the pulldown will appear on the button release and the corresponding menu item will not be marked with an arrow to show it is a cascading pulldown. Do this sort of thing at your own risk

ke

This is the terminator for the command list. If you create pulldowns which add key definitions, aliases, or other menu items, use the same rules for escaping “ke’s” as when you nest key definitions.

Examples

To make a backspace and allow the arrow left to wrap to the previous line for column 1:

From a UNIX prompt enter “ce $HOME/.Cekeys”

At the ‘‘Command:” prompt find the key definition for the Left key by entering:
/kd Left/ and press Return.

The cursor should be on the line:
kd Left   al ke

Insert the characters “–w” to make the line read:
kd Left   al –w ke

Go to the top of the file (key <Ctrl> t) and find the string /kd Backspace/.

The cursor should be on the line:
kd Backspace   ee ke

Insert the characters “–w” to make the line read:
kd Backspace   ee –w ke

Save the file and from the UNIX prompt execute the command:
cd –reload

The new key definitions are now in effect in all windows.

To replace an existing pulldown menu item:

The default menu item for doing text flow is defined by:
mi Edit/5 ’Text Flow’tf ke

This reads, “Define line 5 of the Edit pulldown as displaying the string “Text Flow” and when selected, execute the “tf” ***Ce*** command.” We would prefer that to use the text flow options which do blank precompression “–p” and right justification “–b”. We would also like to use default left and right boundaries of 9 and 80 if the user does not do a rectangular mark over the region to be text flowed. Executing the following line will implement this change. Adding the line to your “.Cekeys” file will make the change permanent.

mi Edit/5 ’Text Flow’tf –p –b –l 9 –r 80 ke

To add a cascading pulldown:
By default, the Pulldown for Macro is empty. We wish to add a cascading pulldown as the first menu item (item 0) which will perform some local functions.

mi Macro/0 ’My Stuff1’pd Mine ke
mi Mine/0 ’Find All’tmw;1,$! –c –m grep –c \\&’Find: ’ ke
mi Mine/1 ’Print File’tmw;1,$! –c –m lp ke
mi Mine/2 ’Print Region’! –c –m lp ke
mi Mine/3 ’Count Words’! –c –m wc –w ke
mi Mine/4 ’Sort Region’! sort ke

This page intentionally left blank

Appendix D: Change History

Introduction

This Appendix summarizes the changes between ARPUS/ce Version 2.5.1 and previous versions. The Appendix is organized in reverse chronological order. That is, changes between the latest revision and its predecessor are listed first. Changes relating to older versions are listed later in this Appendix.

Other chapters of this User Guide contain tables of ***Ce*** and ceterm commands as well as default key definitions. The information shown in this Appendix is also included in those tables for your referencing convenience.

New arguments and X resources

linemax <num>

ceterm.linemax: <num>

Normally a ceterm stores up to 16 million lines of output in the transcript pad. It is more likely you will run out of swap space first. This argument allows you to limit the number of lines for a ceterm. When the limit <num> is reached, the oldest lines are deleted from the top of the transcript pad.
DEFAULT: no limit

Ce.autocut:: { y | n}

This option, when set to “yes”, causes the “ed” (delete), “ee” (backspace), “es” (type a character), “er” (type a special hex character), and “xp” (paste) commands to automatically cut a highlighted area when they are in a key definition by themselves, not part of a list of Ce commands executed under one keystroke. This is similar to Netscape and other PC based editors.
DEFAULT: n

New Ce commands

caps

This command toggles or sets a caps mode. In caps mode, lowercase characters are folded to uppercase. This is useful for working with mainframe files.

New options to existing Ce commands

–w option in the “al”, “ar”, “ee” commands

The –w option to each of these commands causes the cursor to wrap to the previous or next line when the end of line is encountered. Normally “al” and “ee” stop when they end the leftmost column on a line. The –w option causes them to wrap to the rightmost column of the previous line. In the case of “ee”, the new line between the lines is removed, causing the lines to be joined. To make use of this option, modify your .Cekeys file as follows and run “ce-reload”:

```
kd Left         al -w ke        # cursor left 1 column
kd Right        ar -w ke        # cursor right 1 column
kd Backspace    ee -w ke        # delete the character preceding the cursor
```

–w option in the “sic” command

The “sic” command is used when Ce is in “mouse off” mode. In this case, the text and mouse cursors are not tied together and encountered. Normally, “al” and “ee” stop when they end the line and you must click to position the text cursor. The –w option causes the text cursor to go past the end of the line when the mouse cursor is past the end of the line. To make use of this option, modify your .Cekey file as follows and run “ce-reload”:

```
kd m1           sic -w;dr;echo;mouse -on ke
kd m1s          sic -w ; dr;echo -r;mouse -on ke
kd m1u          sic -w;xc ;mouse -off ke;
kd m1us sic -w;xc -r ;mouse -off ke;
#UNIX mode regular expressions
kd *m2 sic -w;/[^-a-zA–Z._@\$0–9\\\\\\\\/\\\\\\~]/dr;?[^-a-zA–Z._@\$0-
#Aegis mode regular expressions
kd *m2 sic -w;/[~a-zA–Z._@@-$0–9@@/@@\~]/dr;[~a-zA–Z._@@-$0–9@@/@@\
```

–f option in the “wc” command

The existing –f option has been modified to close all cc (carbon copy) windows in the file or transcript pad.

Bug Fixes

The following bugs were fixed.

On Linux, a race condition existed in the ceterm terminal emulator in which blocks of data were lost when extensive output was written at the same time to the screen.

In a ceterm window, when the horizontal scroll bar was automatically turned on, it covered the last line of the transcript pad.

Several minor problems involving colored text have been fixed.

The dotmode option on HP/UX terminals is no longer needed due to changes to the pseudo-terminal interface. Dotmode is still supported.

If a file system with hard quotas is active and the file system is over quota, saving the file fails and is not detected. The error no longer appears and then a crash file is created.

Summary of changes from *Ce* release 2.4.2 to 2.5

New Major Features

Scroll Bars

Ce now supports both vertical and horizontal scroll bars. By default, they appear as needed. X resources and commands can be used to control their appearance (both how they look and when they appear). Due to ***Ce's*** ability to scroll sideways, an extra resource "Ce.scrollBarWidth" controls the horizontal scroll bar behavior. This resource can also be used to disable the horizontal scroll bar and keep the vertical scroll bar.

Menu Bar and Pulldowns

Ce now supports a user customizable menu bar and user customizable pull down menus. Menu items, which look a lot like key definitions, are controlled by commands held in the ".Cekeys" file. Several new commands assist in using an manipulating pulldowns.

Colorization

A new ***Ce*** command "ca" allows the colorization of a range of text. The colorization is local to the ***Ce*** session and is not saved. Several programs to generate colorization on HTML and C programs are provides in source along with the menu item definitions which would make use of them. This first color release uses static color. That is, the colorization is not yet language sensitive.

Labels

Ce now allows the generation of labels which are stored local to a ***Ce*** session. If line numbers are turned on, the labels can be seen in the line no field. A program to generate labels from C source and put them in a pulldown is supplied in source.

HTML Documentation

The ***Ce*** helps and users guide are provided in HTML format along with a frequently asked questions list and other HTML only documentation. The HTML package is self contained. You need only provide a link so your intranet web server can get to the main web page.

Contributed Programs

Several useful programs are provided in source form. These programs are designed to interact with ***Ce*** and can server as a model for other programs. The initial set includes a basic HTML parser which generates ***Ce*** "ca" commands to colorize HTML documents; a program which does the same thing for C programs; and a program to generate labels for all the functions in a C source file.

New Arguments and X Resources

–kdp <name>
Ce.keydefProp : <name>

> Ce stored the key definitions it reads from the "$HOME/.Cekeys" file in an X property hung off the root window. This property allows a different name to be used.This option can be used with the –name option or aliasing the ***Ce*** command to maintain multiple sets of independent key definitions for different windows. This falls under the heading of extremely advanced features. It has been used in writing simulators which control bunches of ce and ceterm windows simulating multiple processors. DEFAULT: CeKeys.

–sb {y | n | auto}
Ce.scrollBar : {y | n | auto}

Scroll bars are always on “y” or off “n”, or they appear as needed. As needed is defined as a vertical scroll bar exists when the number of lines being displayed exceeds the number of lines in the window. A horizontal scroll bar is needed when a line is longer than a window. A special note about ***Ce*** horizontal scroll bars: The slider width is an indication of the width of the window with respect to the width of the line. To facilitate horizontal scrolling, ***Ce*** operates as though the line width is at least 300 characters when the scroll bar slider is drawn. This can be modified with the “Ce.scrollBarWidth” option. Note also that the scrollBar option is often set differently for ***Ce*** and ceterm. It is common to turn it off in ceterm windows. Ce.scrollBar : auto ceterm.scrollBar : no DEFAULT: auto.

–sbwidth <num>
Ce.scrollBarWidth : <num>

The argument sbwidth or the X Windows resource “scrollBarWidth” can be set to define the minimum line size used to calculate the dimensions of the slider portion of the scroll bar. For example: If the window is 80 characters wide and one of the lines displayed is 160 characters wide, the horizontal scroll bar slider will be one half the width of the gutter it rides in. This allows you to slide to the end of the line with the slider. The use of horizontal scrolling in ***Ce*** makes this somewhat restrictive. It would be nice to use the horizontal scroll bar to move farther than the end of the line. To facilitate this, the “scrollBarWidth” resource is used. If the horizontal scroll bar is displayed, it behaves as though the longest line displayed is at least “scrollBarWidth” characters long. If a real line is longer, the longer value is used. Setting scrollBarWidth to 1 will make the slider size to the longest line in the window. Setting scrollBarWidth to 0 disables the horizontal scroll bar. This way you can use just the vertical scroll bar without ever seeing the horizontal. DEFAULT: 300.

–oplist

The causes ***Ce*** to calculate it’s run time options and dump them to stdout. It is useful in diagnosing problems and is used in the procedure which updates ***Ce*** from a previous release.

–pdm {y | n}
Ce.pdm : {y | n}

The pdfm or the X windows resource “.pdm” can be set to define whether the pull down menu bar is initially visible or invisible. It is common to have the pulldown menu bar on for ***Ce*** and cv and off for ceterm: ce.pdm : y cv.pdm : y ceterm.pdm : n.

–xrm “resource:value”

The xrm option allows the specification of X resources. This is a standard parameter for X applications. It is not especially useful with ***Ce***, as all the resources can be specified from the command line. The parameter holds one X resource line. The “–xrm” parameter may be specified multiple times. For example: –xrm “ce.foreground:red”.

–CEHELPDIR <path>
Ce.CEHELPDIR : <path>

The CEHELPDIR resource is set in the installation of the app–defaults file for ***Ce*** to the directory containing the ***Ce*** help pages. If you do not set the environment variable “CEHELPDIR” in your “.profile”, ***Ce*** will use this value when looking for the help pages. This technique makes the ***Ce*** help file directory easier to move around.

–dotmode {0 | 1[string] | 2}
ceterm.dotmode : {0 | 1[string] | 2}

The ceterm password dot mode option is needed on HP/UX machines of release 10.20 and beyond where the users primarily use the csh shell. When ceterm detects that the user is typing a password, the program goes into dot mode. The characters typed are displayed as dots and undo processing is disabled so you cannot undo to get the password back. At HP/UX 10.20, it was not possible to find a combination of “tty” settings which would correctly indicate that a password was being entered for the “csh” (ksh works fine). To rectify this, a simpler method of detecting password prompting, namely watching for “pas” in the prompt string (case insensitive), was added. In this mode, you can optionally set the prompt fragment to look for. The values are shown in Table D-1.

Table D-1. Prompt Fragments and Definitions

Value	Definition
0	Nodot mode, never display dots, passwords are visible.
1	Scan prompt for string, Note, if you use hostname in your UNIX prompt and your node name is Pascal, you are sunk!
2	Normal mode, watch the tty bits to determine password entry. DEFAULT: 2 (Can be set during installation).

New *Ce* Commands

New ***Ce*** commands listed in Table D-2 were added to support colorization of text, pull down menus, scroll bars, statement labels, and miscellaneous features people asked for.

Table D-2. New *Ce* Commands and Definitions

Command	Definition
ca	This command is used to color an area or remove color from an area.
{	Called the "corner" command, this command behaves similar to the positioning command [row,col]. Instead of moving the cursor to this position, the corner command positions the file so {row,col} is in the upper left corner of the window.
fbdr	This command allows dynamic inspection and modification of the find border resource introduced in release 2.4. The find border modifies the positioning commands (such as find), to position the window so there are at least 'n' lines between the located position and the found line. In this case, "n" is the number from the "fbdr" command. This allows you to see text in context. See the "fbdr" help for more information. The scroll bar command, "sb", allows the scroll bars to be switched on and off.
mi	The "menu item" command is put in the ".Cekeys" file to create the items found in the menu bar and in pull down menu's. It's syntax is very much the same as a key definition. As with key definitions, menu items are known across all ***Ce*** windows. The local menu item command, "lmi", creates a menu item for just the current window.
pd	The "pulldown" command activates a pull down menu. This command can be placed in a key definition or executed from a menu item. It is the command used to create cascading pull down menus. Try the ***Ce*** command: "pd Menubar".
pdm	The "pull down menu" command allows you to toggle the menu bar on and off.
tmb	The "to menu bar" command can be put under a key definition to move the cursor to the menu bar and lock the main window till the mouse is clicked.

Table D-2. New *Ce* Commands and Definitions (Cont)

Command	Definition
nc	The "next color" command moves the cursor forward to the next colored area.
pc	The "previous color" command moves the cursor backward to the next colored area.
eval	The "evaluate" command performs symbolic substitution of environment variables on its argument and then executes the argument as ***Ce*** commands. This is useful in some complex command files and key definitions.
lbl	The "label" command sets a label at the current cursor position in the main window.
glb	The "go label" command positions the cursor at the named label.
dlb	The "delete label" command deletes all labels in a range of text.
xa	The "paste buffer append" command concatenates two paste buffers. This is useful in some key definitions.
f	Alternate find syntax. Allows delimiter other than "/". This is handy for search strings with slashes in them.

CE_INIT Changes

The Ce_init program is now sensitive to whether Aegis or UNIX regular expressions are in use. The shell now asks several setup questions to do some initial customization. The key definitions, which were previously held in the DATA directory, have been put into the "ce_init" shell. The data directory no longer exists.

CE_ISCETERM

This program sends a terminal control sequence which is ignored by xterm's, hpterms, cmdtools, and so on but is recognized by ceterm. The program can thus detect whether it is being called from a ceterm or some other terminal. This is often useful in ".profile" (.login) shells or ".kshrc" (.cshrc) shells.

CE_INSTALL Changes

The install process now addresses the HTML and contributed program directories. It also allows some site wide customization as part of the install process. The data directory which contained initial key definitions has been eliminated in release 2.5. This data is now encoded in the "ce_init" shell. Also, the help files do not need to be tailored to the site. The only file which is tailored is "ce_init".

Incompatibilities Between 2.5 and 2.4.2

Release 2.4.2 and earlier versions will complain about the menu item definitions which are stored with the key definitions. Once you run release 2.5, bringing up an earlier version of ***Ce*** will cause a bunch of error message to be written out. You can get rid of this by doing a "ce –reload" with the earlier version. The code which generates these messages has been modified so it will not be a problem with later releases.

Changes from Release 2.4.1 to 2.4.2

This is a summary of changes from Ce release 2.4.1 to 2.4.2. Release 2.4.2 contains 2 bug fixes and 2 enhancements.

Bug Fixes

References to the system routine "gethostbyname" were removed to get around a problem which occured on Apollo systems running the xdm window manager.

Correction to allow the below key definitions to work as desired when running ***Ce*** in "mouse –off" mode.
kd m1 sic;dr;echo;mouse –on ke
kd m1U xc;mouse –off ke

Enhancements

If a ***Ce*** session which was piped into is put into read/write mode, a temporary name in "/tmp" is created. Previous behavior was to beep and issue an error message.

A new X resource (titlebarfont) and line option "–tbf" allows the user to specifiy the font to use in the reverse video titlebar at the top of the window.

Changes from Release 2.4 to 2.4.1

This is a summary of changes from ***Ce*** release 2.4 to 2.4.1.

Bug Fixes

Release 2.4.1 is a maintenance release. The following bugs were fixed:

A memory fault occured in ceterm only on HP/UX systems when the prompt was wider than the window and the return key was pressed in the main transcript pad.

The +lineno option on the command line was off by 1. That is, executing "ce +3 .profile" left the cursor on line 4. This is corrected. This option was undocumented in release 2.4.

A problem with “ww –a” has been corrected. It would occationally compress blanks.

A problem in the bl command was fixed. In one case it would find the wrong matching bracket.

BA problem involving the use of aliases within key definitions has been resolved. A key definition which used an alias would only pick up the first command in the alias list.

A problem involving the title command with no options was resolved. The command sequence “title abc;title” would cause a memory fault.

Changes from Release 2.3.2 to 2.4

This is a summary of changes from ***Ce*** release 2.3.2 to 2.4.

New *Ce* Commands

The new ***Ce*** commands are primarily for users who write customized commands and for programmers who write “**Ce** aware” programs. The alias command is a partial exception to this rule as it is used to add default alias names for common commands. For example, find, quit, end, save, and help aliases were added to the “.Cekeys” file created by “ce_init”. To incorporate these changes into your existing “.Cekeys” file, execute the new “ce_update” command.

Table D-3. New *Ce* Commands and Definitions

Command	Definition
alias	The “alias” command allows the user to define new ***Ce*** commands. The syntax is very similar to kd command. New commands may contain symbolic substitution of positional parameter. It is used in the shipped key definition file to add aliases for help, quit, end, and a few other functions.
sp	“sp” is a special command for use with ”Ce aware” programs. A program is “Ce aware” if it is the program executing under a ceterm and it wishes to make use of ***Ce*** features from within the program.
prefix	“prefix” is a special command which modifies the way data being written to the transcript pad in a ceterm is processed. It allows a set of one or more characters to be looked for at the start of each line written to the transcript pad. If a match is found, the rest of the line is assumed to be Ce commands and executed instead of being put in the transcript pads. The ceterm program comes initialized with a default prefix. This allows some shell users to synchronize the current working directory in the ”Command:” window and the shells current working directory in a more elegant way than previously possible.

Table D-3. New *Ce* Commands and Definitions (Cont)

Command	Definition
xl	The “xl” command is “copy literal”. The string “xl” was chosen because it is consistent with “xc” for copy, “xd” for cut, and “xp” for paste. The “xl” command copies the literal string passed as its argument to the named paste buffer. The optional arguments to “xc” apply to “xl”.

New Parameters to Existing Commands

Table D-4. New Parameters and Definitions

Parameter	Definition
–e option to the bang (!) command	The “bang (!)” command causes the passed command to be sent to the shell and executed. The current marked region is passed as stdin to the program/shell script and optionally cut from the file. The standard output from the invoked program is optionally pasted into the cut out spot. The –e option specifies that standard output from the program is a string of ***Ce*** commands to be executed. Note that when used in conjunction with the alias command, you can write your own editor commands in your choice of languages or shell scripts.
Using grave (reverse) quotes (‘) in prompt strings	Normally the processing of a prompt removes leading white space from the response entered in a prompt. Sometimes this is not desirable. Enclose the prompt string in grave quotes instead of single quotes to preserve leading blanks and tabs in the response to the prompt.
–w option to wc	Normally when ***Ce*** returns, it saves the current window size and color scheme. When the next ***Ce***/ceterm starts up, if no geometry was specified via the command line or X resource, it picks up the last one used. The –w option defeats this option. Using the window manager pull down to close a ce/ceterm implies this option.

New Run Time Options and “X” Resources

–dpb <name>
Ce.dfltPasteBuf : <name>

Use the specified name as the default paste buffer. As delivered, this is the paste buffer name CLIPBOARD. It is the default paste buffer name for most Sun applications. The other common name is PRIMARY with is the name used by xterm and other applications. This resource is commonly set by the system adinistator to the value PRIMARY in the app-defaults directory file “Ce”. It is expected that sites not using Sun workstations will change the default to PRIMARY in the app-defaults library to make it easier for ***Ce*** to communicate with xterms and other applications on these machines. DEFAULT: CLIPBOARD.

–findbrdr <num>
Ce.findbrdr : <num>

Use the specified number as a border for find commands. That is, if <num> was specified as 2, make sure there are 2 lines between the top or bottom of the window and the found string. This is ignored if the find is in the first <num> lines of the file. It may be modified if the number of lines in the window is less than double num. This option is often set to 1 or 2 in the “.Xdefaults file”. For example: If the resource “Ce.findbrdr:1” is active, the find command will position the window so there is at least one line of text above or below the found line. DEFAULT: 0.

–ib <path>
Ce.iconBitmap : <path>

This option lets you change the icon used for ce/ceterm. ***Ce*** will use the requested file as the bitmap for the icon. The file should be in the format created by the X bitmap utility. This utility is found in different places on different types of machines. DEFAULT: The standard ce/ceterm icons.

–transpad

This option puts a ***Ce*** window into a special read from stdin mode. It is useful when piping into ***Ce*** from a device such as a hardware monitor which generates data sporadically over a long period of time. When ***Ce*** is running in its normal mode and it is reading from stdin, if you press “^b” (go to bottom of file), the screen will lock until all the data has been read in ensuring that the bottom of the file has also been read. This is fine when using ***Ce*** as a pager for manual pages. When displaying output from some device which is producing messages, a behavior like the transcript pad of a ceterm is desired, only with no UNIX command window. The “–transpad” option provides this behavior. One use for this option has been to provide a monitor window in a debugger. The ***Ce*** program was executed via a “C” popen(3) call and output written to this pipe as needed. DEFAULT: Off.

–offdspl { y | n }
Ce.offdspl : { y | n }

This option overrides a normal "safety" feature of ***Ce*** for use with virtual window managers such as olvwm. Normally ***Ce*** will not allow the specification of a window geometry which has no portion of the window visible. This is because such a window is very difficult to interact with. When using a virtual window manager it may be desireable to bring up ***Ce*** windows in parts of the virtual workspace which are not currently visible. This is especially true during login startup processing. When set to "y", the offdspl argument or X resource allows the specification of geometries which are off screen. Whether this option is required depends upon how the virtual desktop is managed. Managers such as HP/VUE and CDE unmap the windows which are not in the currently displayed desktop. Window managers like olvwm move the window to geometry values which are off screen. It is for this second type of window managers that the offdspl option is supplied. DEFAULT: n.

–lockf { y | n }
Ce.lockf : { y | n }

As of release 2.4, ***Ce*** supports the use of System V, advisory file locking. The default is to use file locking. Specifying "n" in the command line argument or the X resource disables the file locking feature. Note that file locking is only supported on file systems which support advisory locking via the lockf system call. This includes local files systems and most NFS file systems. DEFAULT: Y.

New Executable

ce_update

A new shell called "ce_update" and a supporting program named "ce_merge" have been added. The "ce_update" command compares the existing key definitions to the defaults and appends any new default key definitions which were not previously defined to the existing ".Cekeys" file. Users who customize their key definitions generally do not want to run "ce_init" to get new key definitions because this would overwrite their customizations. The "ce_update" command does a merge without replace of the default definitions and puts the new definitions at the bottom of the file where they are easy to find. Note that "ce_update" is an alias to "ce".

Other Enhancements and Fixes

Enhancements

The abrt command "^x" will not cancel pending prompts. Thus if you press the Help key "F9" and are prompted for a name and you do not wish to look at any helps, pressing "^x" will cancel the prompting command.

The "dq" command (interupt) in ceterm causes the transcript pad to scroll to the bottom of the data. This avoids the problem of users in scroll mode (the default) to continue to see data scrolling on the screen after the quit has taken place. In truth, the data had already been sent from the shell but scrolling one line at a time made the output look like the command was still running. See the "ws" command or the scroll X resource for information about turning scrolling on and off.

The environment variable CE_CMDF_PATH is searched when accessing command files which do not have a complete path. For example, If “CE_CMDF_PATH=.:/usr/mygroup/cecmds:/usr/dept/cecmds” and the ***Ce*** command “cmdf abc” is executed, ***Ce*** will check for the command file as “./abc, /usr/mygroup/cecmds/abc”, and “/usr/dept/cecmds/abc”.

Bug Fixes

The tmw (to main window) command caused the transcript pad to scroll when it should not have.

Key definitions which included entering text and scrolling would cause erroneous data to be displayed. A screen refresh would fix it. An example is: “kd ^1 tl;es’> ’;[+1] ke”.

Several minor drawing, regular expression (substitute), and cursor positioning bugs have been fixed in this release.

CE_INIT Fixes

Moved default “^c” (copy), “^e” (cut), and “^p” (paste) key defintions to “<Alt> c”, “<Alt> e”, and “<Alt> p” respectively.

Added default key definitions for:
“^c” – interupt in ceterm
“^d” – end of file in ceterm

Added aliases for “quit”, “end”, “exit”, “save”, “help”, “print”, “top”, and “bottom”.

Incompatibilities Running Version 2.4.2 with Version 2.3.2:

Several of the changes in release 2.4 make ***Ce*** not integrate properly with release 2.3.2. These changes are described here.

Key definitions using the bang “!” command or the command line prompting, when read from the X server by one version will not be processed correctly by the other. If the “.Cekeys” file was loaded by release 2.3.2, prompts will loose the first character of the prompt and bang commands will lose the first character of the commmand line.

If you use the “Ce.dfltPasteBuf” resource or the “–dpb” option to change the name of the default paste buffer in release 2.4., copy and paste operations using the default paste buffer will not work between 2.4 and 2.3.2 windows.

Changes from Release 2.3 to 2.3.2

This is a summary of changes from ***Ce*** release 2.3 to 2.3.2. It also includes the changes between release 2.3 and 2.3.1. Releases 2.3.1 and 2.3.2 are maintenance releases. The primary changes deal with problems encountered with the license manager introduced in release 2.3. There are several minor bug fixes and two new resources.

Ce.envar: {y | n}
Release 2.3 introduced the ability to click on a file which contains environment variables and

have those variables resolved before looking for the file name. In release 2.3.1, a resource was added to allow the processing of these names to be turned off.

Ce.LSHOST : <nodename>
Release 2.3.2 introduced the ability to specify a default value for the license server LSHOST environment variable as a resource. This allows the system administrator to specify it in the app–defaults/Ce file instead of having to put it in each user's .profile.

Stored cp commands:
If a key definition contains a 'cp' command (none of the default definitions do) and it is read from the .Cekeys file by release 2.3.1 of Ce or earlier, release 2.3.2 will not be able to extract it when starting up.

Changes from Release 2.1 to 2.3

This is a summary of changes from Ce release 2.1 to 2.3. It includes the changes between release 2.1 and 2.2.

New Ce Commands

Table D-5. New *Ce* Commands and Definitions

Command	Definitions
bell	Beep the bell
mouse	Disconnect Mouse and text cursors
rec	Command recording
title	Change Window Manager title

New Parameters to Existing Commands

Table D-6. New Parameters and Definitions

Command	Option	Definition
pn	–r	Replace target file name if it exists
tf	–p	Compress blanks before textflow
xc	–a	Append mode for copy buffers
vt	–auto	Switch to VT100 emulation mode as needed

New Run Time Options and ”X” Resources

Table D-7. New Run Time Options and “X” Resources and Definitions

Option	Resource	Definition
–iconic	Ce.iconic	Start Ce as an Icon
–ls	ceterm.loginShell	Cause .profile to be run during startup
	ceterm.man	Eat vt100 control sequences when not in vt100 mode
–readlock		Disable “ro” command for a ce session
–vt	ceterm.vt	Control vt100 emulation mode

Other Enhancements

- The “.Cekeys” file now allows the “cmdf” command to be present. This provides an include facility for the “.Cekeys” file. The only commands allowed in the “.Cekeys” file and the included files are “kd”, “cmdf”, “wdc”, and “wdf”. This facility allow group and system wide key definitions to be included from the “.Cekeys” file.
- The “ce”, “cv”, “pn”, and “cmdf” commands allow environment variables to be used in path names. The environment variables correspond to the environment at the time ***Ce*** was invoked. The “env” command can be used to examine and set environment variables within the ***Ce*** session.
- The default key definitions as installed by “ce_init” have changed for the actions of the mouse. The default M1 does a highlighted drag with a copy on the upstroke and M2 does a paste. The former M1 functionality has been moved to M3. Help has been moved to “F9”. Shift and control of the mouse keys has been added. See the “.Cekeys” file created by “ce_init” for more information.

Note to System Administrators (Installers)

The default key definitions in the “data” directory are now delivered in both Aegis and Unix regular expression mode. The Unix mode definitions are commented out. You may modify the data files to initially define the Unix mode definitions. See the keyCon help file for more information on doing this customization. If this change is made, you need to modify the “Ce.expr” line in the “.Xdefaults” file of the “data” directory to specify Unix mode expressions. The location of the data directory is set at install time.

The key definitions for “F2” through “F4”, “^c”, “^e”, and “^p” have been updated in the “initial .Cekeys” file to remove the “–l” (local) option from the definition. You should make this change in your personal “.Cekeys” file. The “–l” option prevented copy and paste across windows from working.

For example:

OLD:	kd F2 xc –l ke
	kd F2S xc –l –r ke
NEW:	kd F2 xc ke
	kd F2S xc –r ke

In the files used by “ce_init” to build the “.Cekeys” files, alternate Unix mode definitions have been provided. These definitions are initially commented out. They can be updated by the user in their private “.Cekeys” file or by the system administrator in the master copies. In the “keydef” file (either “.Cekeys” or the master copy), find the line “Ce.expr”. It will be in a comment line. Below it will be two key definitions. One is commented out with the string “#Unix expr”. Delete this string and comment out the existing definition on the line above.

Incompatibilities Running Version 2.3 with Version 2.1

Several of the changes in release 2.3 make ***Ce*** not integrate properly with release 2.1. These changes are described here.

- Default Saved Paste Buffer name:
  When a ***Ce*** session is ended, ***Ce*** takes the paste buffers it owns and copies them to files in the paste buffer directory. By default this is “~/.CePaste”. This permits other ***Ce*** sessions to paste from this buffer after the process has terminated. Special code originally saved the default paste buffer as name “.UNNAMED” in the paste buffer directory. This code has been removed and the name used now is “CLIPBOARD”. The result is that if a release 2.2 or 2.1 terminates, a release 2.3 ***Ce*** will not be able to see the saved paste buffer. The same is true if a 2.3 terminates.
- To avoid problems which occurred when many windows were brought up during login processing and to support the new paradigm for cc window processing, changes were made in the way ***Ce*** windows make their presence known to each other. The result is that ***Ce*** 2.2 and earlier windows will not see ***Ce*** 2.3 windows when searching to see if a window is already open on that file. Also, the “tn” and “ti” commands will not work between ***Ce***, ceterm and cc windows which are of different revisions.
- Processing for the “wdf” and “wdc” commands have changed for release 2.3. Thus release 2.2 and 2.1 windows will be unable to use the wdc and wdf lists created by 2.3 ***Ce***. The reverse is also true.

Appendix E: Installation Summary

This Appendix contains explicit instructions on how to install the ce software from:

- media
- the Internet
- a backup directory containing ***Ce*** installation tar files

Installing *Ce* From Media

Your ***Ce*** license may allow the use of the software on several different platforms. For each licensed platform, the media contains one tar file that contains the entire ***Ce*** package for that platform. Each tar file on the media ranges in size from 2 to 4 MegaBytes.

Instructions

1. The “ce_backup” directory.
   Select a “backup” directory into which the entire media contents can be restored. Ideally, it will be a directory on a host which can be reached by other hosts on which the software is to be installed. We’ll call that directory “ce_backup”.

2. Log on to a system with which you can restore the ***Ce*** tar files from media. You may have to be logged in as root.

3. Set your directory to “ce_backup”.

4. If the media is on tape, place the media in the drive and execute these commands:

   UNIX: mt –f <device> rewind

   UNIX: % tar xvf <device>

5. If the delivery media is a floppy disk it is in pc format. Put it in a suitable machine and copy the file to the “ce_backup” directory. The file name on the diskette is ce.tz. Name the file ce.tar.Z in the “ce_backup” directory. The file will be in a compressed format. Uncompress the file in the “ce_backup” directory using the command:

   Command: % uncompress *.tar.Z (can use compress –d)

6. Now proceed to the *Installing **Ce** from a Backup Directory* Section of this Appendix.

You should only have to perform the above steps once. Once the files have been restored to “ce_backup”, they can be used for all subsequent installs onto any hosts that can reach “ce_backup” in some way (ftp, rcp, NFS, ...).

Installing *Ce* from Data Downloaded from the Internet

The file downloaded from the internet is a compressed tar file. A separate compressed tar file is available for each machine type. Only the binary files (in the bin directory) are different between machine types.

Instructions

1. The “ce_backup” directory.
   Determine the location of the “ce_backup” directory. This is a directory into which the ce tar files have been retrieved from the Internet. The machine type of the machine being installed on should match the machine type of Ce. That is: Install HP/UX binaries on HP/UX machines. The tar file will be referred to as “ce_tar_file”.

2. Log on to a system with which you can read and write the ce tar files.

3. Set your directory to “ce_backups”.

4. Execute the command:

   Command: %compress –d ce_backup/ce_tar_file.Z

5. Now proceed to the *Installing **Ce** from a Backup Directory* Section of this Appendix.

Installing *Ce* from a Backup (TAR) Directory

These steps should be repeated once for each host on which you are installing the software. Note that you can often install on one node and propagate files and links to other nodes using network management software. You can also install into directories which are visible to many nodes via NFS, DFS, or AFS. The total installation time is five minutes.

Instructions

1. The “ce_backup” directory.
   Determine the location of the “ce_backup” directory. This is a directory into which the ce tar file which was restored from media or the Internet will be extracted from tar format.

2. Execute the command ”tar xvf ce_tar_file”. This will create the directory ***Ce*** in the current directory.

NOTE: If you are running on a Domain/OS system, you must do the following step from a SYS5 shell.

3. Select and create, if necessary, the directory where you will want the ***Ce*** binary files to be stored (e.g. /usr/ce/bin). The binary files are “ce”, “xdmc”, “ce_init”, “ce_report”, “ce_update”, and “ce_isceterm”. Soft links ceterm, cv, and ce_merge are also created in this directory. These files occupy about 2.5 megabytes of disk storage.

4. Select and create the directory where you will want the ***Ce*** help files to be stored (e.g. /usr/ce/help). The help files occupy about 2 megabytes of disk storage.

5. Select the directory where you will want the “Ce app–defaults” file to go. This should be somewhere the users “XFILESEARCHPATH” will pick it up. This is a small file.

6. Select the directories where you will want the manual pages to be placed. This should be somewhere the users “MANPATH” will pick it up. These are small files. The manual pages ce.1, xdmc.1, and ce_isceterm.1. Soft links ceterm.1 and cv.1 are also created.

7. From the “ce_backup” directory, execute the command “ce/bin/ce_install”. “ce_install” is a script that performs the installation. It will prompt you for the names of the directories you set up in previous steps and them move the files to their appropriate locations. In addition, if the software is shipped with the network license software the license server (lserv) may be started as part of the install script. See the section “File List” and “Installation Considerations” for information on the directories. Note that if ce already exists in the target directory, it is renamed to ce.r”version” where “version” is the version number of that release.

8. On systems other than SunOS 4.1.* and Apollo domain OS, the ceterm program must be setuid’ed to root. This is required to open a pseudo-terminal and update the utmp information. The setuid in Ce is dropped as soon as these functions are completed or determined to not be needed. It is possible to install Ce without being root using the –noroot parameter to ce–install. Running the installation with “ce/ce_install –noroot” will install the files assuming you have write permission to the target locations. If you use this technique, you may need to get someone with root authority to setuid the executable before you run ceterm.

9. The HTML documentation can be downloaded from the Internet or was provided on installation media. at the anonymous FTP site, the HTML documentation is in a file named cehtml25.tar.Z. On PC format floppy diskette, it is named cehtml.tz Copy the file to a location accessible from your web server and name the file cehtml25.tar.Z Then uncompress the file with the UNIX command:

   compress –d cehtm.25.tar.Z

   At this point, you have a UNIX tar format file. When expanded by the tar command, this file will produce a directory in the current working directory named ce. Change directories to where you want the HTML documentation to be placed and execute the UNIX command:

   tar xvf cehtml25.tar

   From some suitable web page on your site, create a link to the web directory “/ce/ce.html” to attach the ce documentation to your intranet. Note that the HTML files in the documenation are self contained and need no customization.

User Installation

These steps should be repeated for each user of the ***Ce*** software and are contained within the ”new_messages” file for mailing to new ***Ce*** users.

Instructions

1. If the software includes the network licensing software then the environment variable “LSHOST” should be set in the user’s “.profile” or “.login”. “LSHOST” must be set to the host name of the License Server (lserv). For example, the .profile might include:

   LSHOST = host1; export LSHOST

   Where “host1” is replaced by the name of the node where the license server is currently running. Before continuing with the user installation the user should ensure that “LSHOST” is set by logging out/in again or manually setting it.

   An alternative which does not require modifying the users environment, is to set the “Ce.LSHOST” X resource in the “/usr/lib/X11/app–defaults/Ce” file created by the install process.

2. The user should enter “ce_init” which will create the following files in the “$HOME” user’s home directory:

   $HOME/.Cekeys (key definitions)
   $HOME/ce_practice (practice file)

   In additions, the user will be placed within ***Ce***, editing the file ”ce_practice”.

File List

The ***Ce*** installation consists of a set of files that are accessed by all ***Ce*** users and a set of files for each individual user.

Ce consists of the directories and files listed in Table E-1. File and directory names in **boldface** refer to objects that all users can access. File and directory names in *italics* refer to files only accessible by the system administrator. “site_installation_path” is the main directory that contains the ***Ce*** software and its location is selected by the system administrator at the time of installation.

Table E-1. *Ce* Directories and Files

bin		commands and scripts	
	ce	***ce*** executable	(Note 1)
	ce_deinstall	de–installation script	
	ce_init	user initialization script	(Note 1)
	ce_install	installation script	

Table E-1. *Ce* Directories and Files (Cont)

	cv	link to ***ce***	(Note 1)
	ce_isceterm	check for ceterm in .profile	(Note 1)
	ce_report	bug report script	(Note 1)
	ce_merge	link to **ce**	(Note 1)
	ceterm	link to ***ce***	(Note 1)
	ce_update	*user update script*	(Note 1)
	license_key	Licence key for Ce from ETG	(Note 2)
	lserv	license server (not all plat- forms)	(Note 2)
	lsmon	license server monitor	(Note 2)
	xdmc	utility to send ce commands	(Note 1)
help		help file directory	(Note 3)
	< **numerous help files** >		

NOTES:

1. Moved to the binary directory specified in the install script. These files are used a lot and should be copied local to each node.
2. Moved to the license server directory specified in the install script. These files only exist on some machines.
3. Moved to the help directory specified in the install script or the manual page directory specified in the install script. Used when the user requests help.

Installation Considerations

This section describes location dependencies between ***Ce*** files.

The actual ce/ceterm/cv executable has no location dependent information and can reside anywhere. As it tends to be heavily used, having it installed locally on a node can be an advantage. Some platforms page the text portion of executables from the actual executable file and paging across the network is slower than local paging. Also, a network hit will bring ***Ce*** down with no chance for ***Ce*** to do any data preservation.

The “ce_init” shell script has references to the help directory and also to the ***Ce*** executable. If the help directory is moved or the ***Ce*** executable is moved, this file will have to be edited. The “ce_report” shell also references the help directory. This shell script will have to be modified if the help directory is moved.

The help directory is a good candidate for an NFS mount point to a centralized location. The help files are useful are useful but are not used as much as they should be.

As of release 2.5 the data directory is no longer used.

The key definitions are installed by default using Aegis expressions. If the site default is to be Unix, the system administrator should specify this in the installation. The users can override this when they run ce_init.

Appendix F: License Server

As of release 2.3, ***Ce*** is sometimes delivered with a license server. This is LicenseServ by Viman software. ***Ce*** uses a floating license scheme with license sharing based on X display. Thus, you can have any number of ***Ce*** windows open on your node and still be using one license. This includes ***Ce*** sessions running on other nodes which are displayed on your workstation. **The license server is not supplied on all platforms!** If the installation for a given machine type does not include the lserv and lsmon executables, they are not used.

Client (*Ce*) Processing

When ***Ce*** starts up, it contacts the license server by making a call (using UDP/IP for network communication) to request permission from the server to commence execution. To obtain the hostname of the machine on which the server is running, ***Ce*** first checks the environment variable “LSHOST”. If “LSHOST” is not set, it looks for a file named “lshost” in the current working directory. On the RS6000, the file name “LShostname” is used. If the server name is not specified through these two methods, an automatic broadcast detects any license servers on the subnet. This automatic detection keeps the network licensing of ***Ce*** transparent to users.

As an alternative to having the environment variable “LSHOST” set on each users node, ***Ce*** allow the specification of the “LSHOST” value in the “app–defaults” file using resource name “LSHOST”. The normal location for the application defaults resource file for ***Ce*** is “/usr/lib/X11/app–defaults/Ce”. By setting the “LSHOST” resource in this file to the node the license server resides on, the users environment does not have to be modified. If the “LSHOST” variable is specified in the users environment it will take precedence over the value in the application defaults file.

SERVER (lserv)

The installation process puts the license server program in the directory requested at installation time. It has the name “lserv”, it may be moved to the machine the license server will run on. Also included in the distribution is a program called “lsmon”. It can be used to check on license usage.

The license server program is tied to a specific machine on the network. It can be started up automatically when the machine boots up, or it can be started manually. On start up, the server process reads an encrypted initialization file, which gives it information about all features licensed by LicenseServ on this network. The file path name can be specified as a command–line argument to the server start-up command. If the file is not found, it is an error. A single encrypted text string (licensing code) is contained in this file for each license feature, and each feature is treated independently of the others handled by the server process. Each encrypted licensing code includes the name of the feature ***Ce*** and the number of authorized concurrent copies of the feature. This licensing code is obtained from ETG Inc. Customer Service at the following location.

PHONE: 770–64–ARPUS (404–642–7787)
FAX: 770–993–4667
EMAIL: arpus@etginc.com
ADDRESS: ARPUS Customer Service
Enabling Technologies Group, Inc.
8601 Dunwoody Place
Suite 300, Atlanta, GA. 30350

The server always listens to the network traffic on port number 5093. All applications licensed by LicenseServ use this port number to communicate with the server. Site administrators should ensure that this port is not used by other applications. The following line should be added to the "/etc/services" file to indicate that this port is being used by LicenseServ.

LicenseServ 5093/udp

NOTE: You cannot change the port number used by LicenseServ by changing the entry in "/etc/services".

Starting the License Server

The license server can be started at boot time or manually. It is expected that the full path name will be specified. You do not have to be root to run the server, but you must have read permission on the <LicenseCodeFile>. The license server command structure is shown below:

./lserv [–s/Licenserc] [–l/logfile] [–q]

"–sLicenserc" is the name of the file containing license to be used by this server. Note that licenses (other than demo licenses) lock the server to a certain IP address which you supply to ETG Inc. when you request the license key. The default name is Licenserc in the current working directory. The installation script puts the key file in the same directory as the lserv binary with the name License_key. For example, "/etc/lserv –s/etc/License_key –l/usr/spool/lserv.log", might be put in an rc.local startup script for the license server machine. This assumes the installation script was instructed to put the license server in "/etc".

"–l/logfile" specifies where the license server will put its log file. If the "–l" argument is not supplied, the license server checks the environment variable "LICENSESTAT" to obtain the name of the file. The default is not to create a log file.

The argument "–q" suppresses message generation.

The log file message format is shown in figure F-1.

The code 0 is used for Issues, 1 for Denials, and 2 for Returns. Key-life is the total time the application used its key, and is recorded only for Returns.

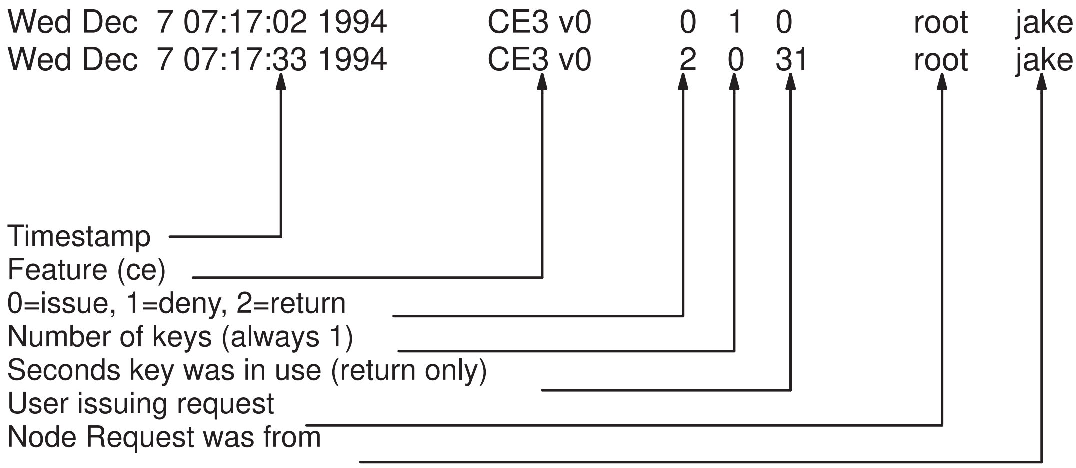


Figure E-1. Log Message Format

Suggestion

Get a separate license key set from ETG for each subnet and start one copy of the license server on each subnet.

Monitoring License Usage

The lsmon utility is used to retrieve information about all features currently licensed by the LicenseServ server, and information about clients active on the network.

To run lsmon, use the following command at your Unix prompt:

```
% lsmon [hostname]
```

If hostname is omitted, lsmon tries to use the environment variable “LSHOST” to obtain the name of the LicenseServ server . If that fails, lsmon looks at the contents of the “lshost” file in the current directory. If that also fails, an automatic broadcast detects any license servers on the subnet.

End of Document

This page intentionally left blank