

---

# UNIX Basics

Science & Technology Support Group  
High Performance Computing  
Ohio Supercomputer Center  
1224 Kinnear Road  
Columbus, OH 43212

# Table of Contents

---

- [UNIX Basics--Introduction](#)
- [Useful Concepts](#)
- [Useful Commands](#)
- [Files and Directories, Part I](#)
- [Screen Editors](#)
- [Files and Directories, Part II](#)
- [Special Characters](#)
- [UNIX Shell--Basics](#)
- [References](#)
- [Exercises](#)

# Purpose of Course

---

- Introduce the basics of UNIX
- Introduce the basics of UNIX screen editors
- Provide hands-on practice

# Acknowledgments

---

- **Course notes are based on:**
  - UNIX Primer Plus, Third Edition
    - Don Martin, Stephen Prata, Mitchell Waite, Michael Wessler, and Dan Wilson
    - Waite Group Press, A Division of Macmillan USA, Inc., 201 West 103rd Street, Indianapolis, Indiana, 46290 USA © 2000
- **Science and Technology Support Group, OSC**
  - Leslie Southern
  - Troy Baer
  - Scott Brozell
  - Peter Carswell
  - Dave Ennis
  - Jim Giuliani
  - Elaine Landwehr
- **Brian Powell**

# UNIX Basics--Introduction

---

- **What is UNIX?**
  - An operating system **and** attendant applications programs
- **Why use UNIX?**
  - Available on virtually all machines in one format or another
  - Long history
  - Has been adapted to new platforms
- **On what is UNIX based?**
  - Uses C language

# UNIX Basics--Software

---

- **Operating**
  - for the computer
  - liaison between computer and user
- **Application**
  - for the users
    - electronic filing
    - word processing
    - database maintenance
    - electronic mail and networking access

# UNIX Basics--Structure

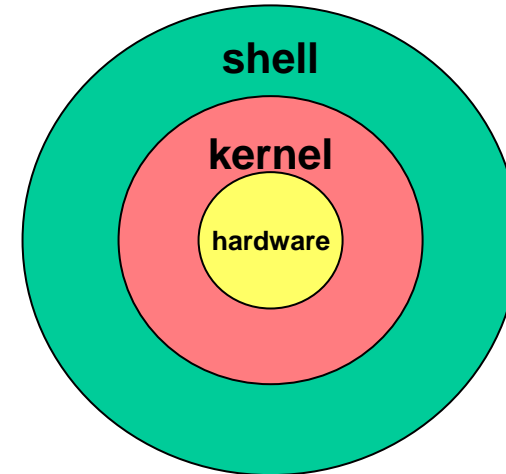
---

- **Kernel**

- CPU scheduling
- memory management
- process management
- other duties
- not for average users

- **Shell**

- interacts between kernel and user
- user invokes commands
- types
  - Bourne shell
  - Korn shell
  - C shell
  - TCshell
  - BASH



# Useful Concepts

---

- [Getting out of Trouble](#)
- [Terms and Concepts](#)

# Getting Out of Trouble

---

- **Control c**
  - stop a process
- **kill**
  - kill a job in the works
- **Control u**
  - delete a full line to the prompt
- **Control u**
  - undo last command
- **Control s**
  - stop scrolling
- **Control q**
  - resume scrolling
- **What happens if you type a word that is not a command?**

# Terms and Concepts

---

- **Standard input**      **stdin**
  - keyboard
- **Standard output**      **stdout**
  - print on the screen
- **Case sensitive**
  - it matters whether you use uppercase or lowercase
  - UNIX commands--usually lowercase
- **Control key**
  - if it is part of a command, the control key and the second key are pressed simultaneously
- **Return key**
  - almost always used to tell system you have finished typing the command

# Terms and Concepts

---

- **prompt**
  - a symbol (usually % or \$)
  - have the ability to change your prompt (later)
  - when cursor is at the prompt, you can enter a command
- **Permission**
  - means ability to read, write, or execute a directory or a file, based on user, group, and other categories

# Useful Commands

---

- [Structure](#)
- [First \(and Last\) Commands](#)
- [Easy Commands](#)

# Structure

---

- **Command structure**
  - `command -option argument`
- **command**
  - usually lowercase
  - what you want to do
- **-option**
  - sometimes not required
  - enhances output of command
  - tailors output to your needs
  - can be combined with one or more other options
- **argument**
  - what your command will act upon
  - can have more than one argument
  - sometimes not required

# First (and Last) Commands

---

- **Logging on**

- `userid`

- assigned by systems administrator
    - probably won't change

- `password`

- assigned by systems administrator
    - should not share it

- `passwd`

- use `passwd` command to change your default password to one you like
    - should change your password from time to time for security
    - on OSC systems: `oscpasswd` changes password on all machines

- **Logging off**

- `exit`

- this is usually the way to log off
    - may differ from system to system

# Easy Commands

---

---

Command	Options	Arguments
date	[no options]	[no arguments]
cal	[no options]	<i>year</i> or <i>month year</i>
finger	<i>[-m, -l, -s]</i>	[name]
help		
man		[argument]
man	<i>-k</i>	[argument]
who	[several options]	
who am I		

**NOTE: end commands with a return**

---

---

# Easy Commands--Try This

---

type

```
date
```

type

```
cal year_you_were_born
```

type

```
who
```

type

```
finger woodall
```

type

```
finger -s woodall
```

type

```
finger -s woodall
```

***NOTE: end commands with a return***

---

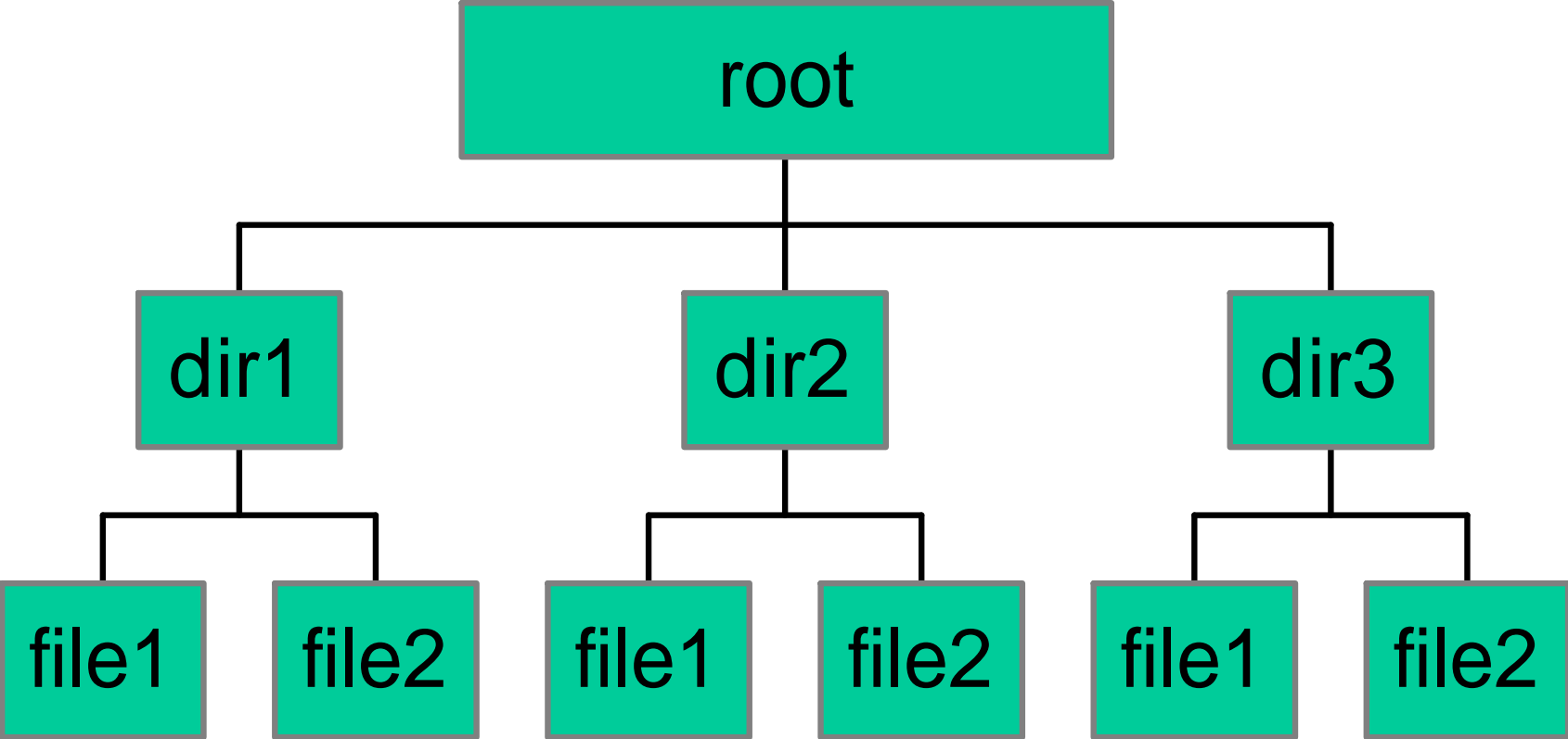
# Files and Directories, Part I

---

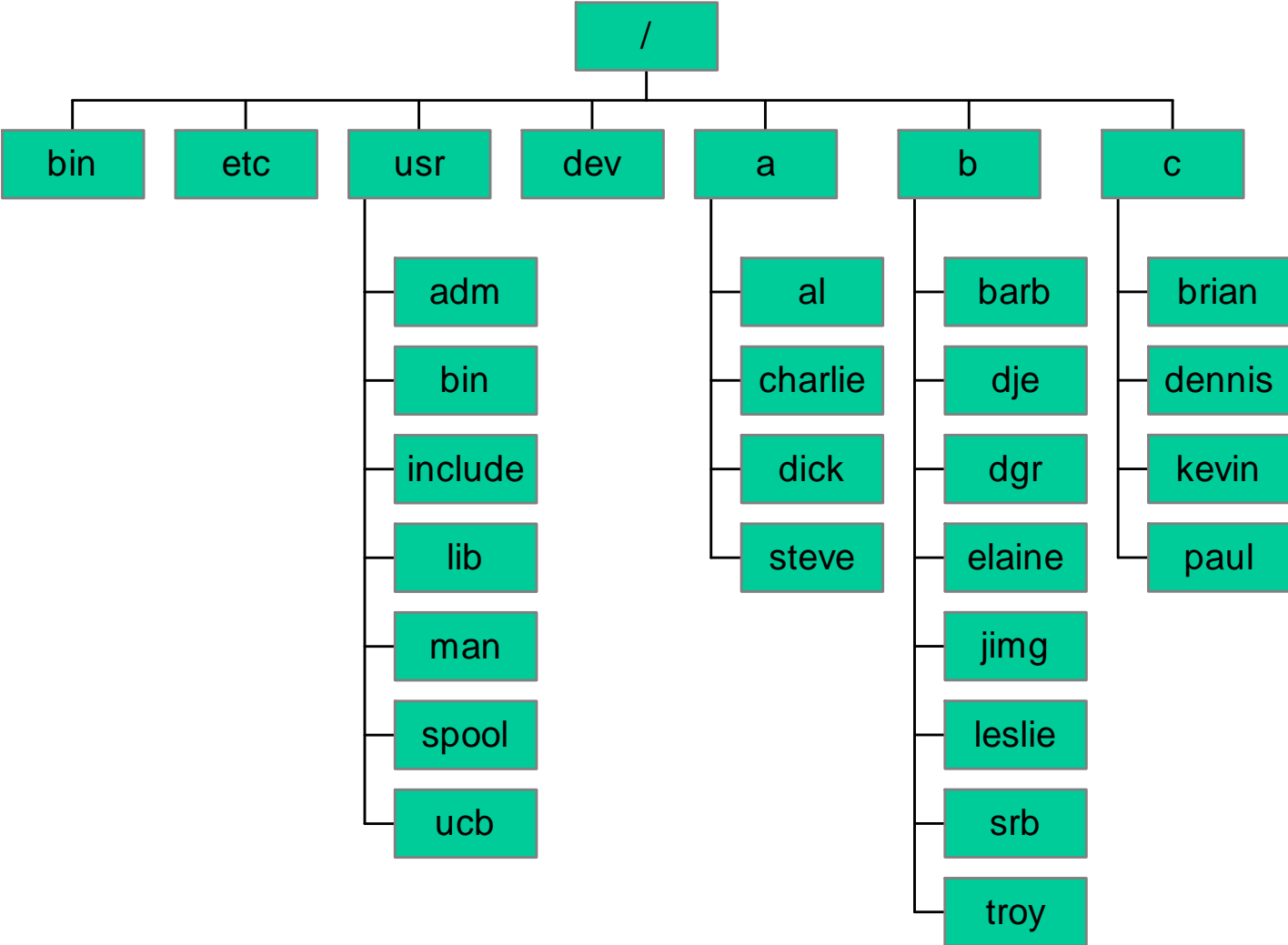
- [File Hierarchy](#)
- [Where Are You](#)
- [Files and Directories--First Commands](#)
- [Files and Directories--Naming](#)
- [Files--Creating](#)

# File Hierarchy

---



# File Hierarchy



# Where Are You

---

## `pwd`

- tells you where you are in the file hierarchy
- important in increasingly complex hierarchy
- important for moving files
- important in moving to and from directories of others

# Files and Directories--First Commands

```
$ ls -a -c -l -f -r -s -t directory -R
```

```
$ cat [several] file
```

```
$ more [several] file
```

```
$ lpr -l[others] file
```

```
$ rm file
```

# Files and Directories--Try This

```
$ ls
```

```
$ ls -a
```

```
$ ls -l
```

```
$ ls -F
```

```
$ ls -r
```

```
$ ls -R
```

```
$ ls -al
```

# Files and Directories--Naming

---

- avoid spaces; separate words with dots or underscores
  - **my.file**
- begin directory names with capital letters
  - **My.directory**
- avoid special characters
  - / \ ' " ; - ? [ ] ( ) ~ ! \$ { } < >
- make names descriptive
  - **letter**
    - not particularly descriptive
  - **woodall.010403.letter**
    - more information
- avoid ending file name with a number
  - **can cause problems in converting files, depending upon your system**

# Files--Creating

---

```
cat > file
line
line
line

control d
```

now type

```
more file
```

- using **>** to redirect output from a command to a file
  - *command > file*
  - *more file*

```
cal > calendar
more calendar
```

# Files--Creating--Try This

```
$ cat > workshop
first line
second line
third line
control d
$ more workshop
$ cal > calendar
$ ls
$ more calendar
$ more workshop
$ cal > workshop
$ ls
$ more workshop
$ ls
$ rm workshop
$ ls
```

Do you have a file  
called workshop?

# Screen Editors

---

- [vi](#)
- [emacs](#)
- [Other Editor--Pico](#)

# vi editor

---

- early form of word processing
- not especially easy to use
- **HOWEVER--exists on all UNIX systems**
- good to know, even if you use different editor
- approximately 100 commands

# vi--Opening a File; Modes

---

- **open a file**
  - `vi file`
- **modes**
  - input
    - different ways to get into input mode
    - only one way to get out of input mode: escape key
  - command
    - always in command mode when you open up a file in vi

# vi--First Commands, Part I

---

- **open a file**
  - `vi file`
  - `vedit file` (bottom line will show when you are in insert mode)
- **move the cursor**
  - `h` (left), `j` (down), `k` (up), `l` (right)
- **enter insert mode**
  - `a` (append after cursor), `A` (append at end of line), `i` (insert at cursor), `I` (insert at beginning of line), `o` (opens a line below the cursor), `O` (opens a line above the cursor)
- **leave insert mode**
  - Escape key
- **delete or replace**
  - `x` (delete character), `dd` (delete line), `r` (replace one character, do not enter insert mode)

# vi--First Commands, Part I--Try This

---

- `vedit practice`
- **watch bottom right of screen**
- **i**
  - What happens
- **Escape**
  - What happens
- *i characters*
- **move in and out of insert/append/open mode**
- **look at the differences between the various ways of going into input mode**
- **type some lines, press Escape, use h, j, k, and l to move cursor**

# vi--First Commands, Part II

---

- **undo**
  - `u` (last command)
- **save and quit the editor**
  - `zz` (save the changes and quit), `:w` (save the changes, do not quit), `:q!` (quit, do not save the changes), `:wq` (save the changes and quit)
- **erase**
  - delete key
  - sometimes need to use Control h combination
- **set line wrap**
  - only for current editing session; must reset each time you open the file
  - `:set wm=15`

# vi--First Commands, Part II--Try This

---

- type a few characters; press Control u
- type a line that goes beyond the line on the screen; move cursor up; note that this is one line
- go into command mode, type `:set wm=15`
- type another long line
- close the file; reopen it; type another long line
  - What happens?

# vi--Additional Commands, Part I

---

- **position cursor**

- Control **d** (scrolls cursor down, usually 12 lines), Control **f** (scrolls the cursor forward, usually 24 lines), Control **b** (scrolls the cursor back, usually 24 lines), Control **u** (scrolls the cursor up, usually 12 lines); **e** (end of word), **b** (beginning of word), **G** (end of document), **nG** (go to line *n* of the document)
- Control **g** (get line number where cursor is)
- **/pattern** (search forward for pattern), **n** (search for next instance of pattern), **?pattern** (search backward for pattern)
- **§** (from the cursor to the end of the line), **o** (from just before the cursor to the beginning of the line), **)** (from the cursor to the beginning of the next sentence), **(** (from just before the cursor back to the beginning of the sentence where cursor is), **}** (from the cursor to the end of the paragraph), **{** (from just before the cursor back to the beginning of a paragraph)

# vi--Additional Commands, Part I--Try

---

- open your document
- move your cursor around
- insert more text
- save, exit, reopen
  
- search for a character, e.g., a letter e; find the next e
- search backwards for a character; find the next earlier occurrence

# vi--Additional Commands, Part II

---

- **operate on words, lines, paragraphs**
  - **c** (change word or words as indicated and enter insert mode), **d** (delete items as indicated and store in buffer for possible placement elsewhere), **y** (copy lines as indicated and store in buffer for possible duplication elsewhere)
    - abbreviations for words, sentences, lines, or paragraphs: **w**, **b**, **e**, **c**, **(**, **)**, **0**, **\$**, **{**, **}**
- **print storage buffers**
  - **p** (print buffer contents after the cursor), **P** (print buffer contents before the cursor)
- **join lines**
  - **J**
- **change case of letters**
  - **~** (a toggle)
- **check spelling**
  - `spell file` or `spell file | more`

# vi--Additional Commands, Part II--Try

---

- **position cursor at beginning of a word**
  - `cw`
  - what do you see?
  - type some characters
  - Escape
- **position cursor in middle of a word**
  - `cw`
  - what do you see
  - type some characters
  - Escape
- **position cursor at beginning of a word**
  - `2cw`
  - what do you see
  - type some characters
  - Escape

# emacs

---

- free-to-download
- widely available
- more modern
- online help
- online tutorial
- no modes
- document has prompts for user

# emacs--Opening a File; Information

---

- **open a file**
  - `emacs file`
- **information areas**
  - echo area
    - displays certain commands
    - prompts you for input to a command
    - when cursor is in echo area, can use any emacs editing tools that work on one line to change what you have typed
      - to abort a command started on the echo area: Control g
  - mode line
    - can be ignored when working on simple text editing on single files

# emacs--First Commands

---

- **position cursor**
  - Control **p** (up), Control **n** (down), Control **b** (left), Control **f** (right)
  - (may be able to use arrow keys, depending on how your terminal is set up)
- **position cursor, additional**
  - Control **a** (beginning of line), Control **e** (end of line)
- **position cursor, with numerical arguments**
  - precede numerical argument with an Escape key
  - examples
    - Escape **4** Control **p** (move up 4 lines)
    - Escape **5** Control **f** (move forward to 5th character)
    - Escape **8** Control **n** (move down 8 lines)

# emacs--Basic Editing

---

- **add text**
  - move cursor to the correct position and start typing
- **delete text**
  - Control **d** (delete character at cursor), Delete key (delete character before the cursor), Escape Delete (delete word before the cursor), Escape **d** (kill the word after the cursor), Control **k** (kill from cursor to end of line), Control **y** (yank back a previous kill)
  - concepts
    - delete: not meant to use deleted characters again
    - kill: killed characters go into a buffer; limited capacity; deletes oldest kills when new ones are added

# emacs--Basic Editing

---

- **cut and paste**

- kill and yank--the usual ways to cut and paste
- commands

- Escape **d** (kill one word), Control **n** (move the cursor), Control **k** (kill one line), Control **y** (yank back the killed line), Escape **y** (replace yanked-back line with killed word), Escape **Y** (replace killed word with previous kill), Escape **Y** (replace previous kill with next previous kill, etc.)

- **undo changes**

- Control **x u** (undo last change), Control **x Control c** (leave without saving--prompt appears), Escape **x** (restore buffer to original contents)

- **conserve CPU time**

- in documents with text already entered, insert a couple of blank lines and type there (prevents screen from having to redraw with the addition of every character)

# emacs--Basic Editing--Try This

---

- **emacs workshope**
  - identify the echo line
  - note the changes and information in echo line as you type
  - identify the mode line (in reverse type)
- **type some lines**
- **move the cursor--practice using key combinations, as well as using the arrow key**
- **move cursor with numerical arguments**
  - Escape 4 Control p
- **practice deleting a character, the rest of a line, restoring a deleted character**

# emacs--Basic Editing

---

- **manage line length**
  - press Return at end of line
  - use `auto-fill mode`, for current editing session only
    - Escape `x` `auto-fill-mode` (toggle auto-fill), Escape `64` Escape `x` `set fill-column` (set line length to 64 characters), Escape `64` Control `x` `f` (alternative method of setting line length to 64 characters)

# emacs--Commands

---

- **more than 400**
  - have long names
  - can have abbreviations
  - abbreviations are bound to a command
  - for abbreviations, refer to command dispatch table
- **long name use**
  - Escape **x** *command-name*
- **long name use--workarounds**
  - keystroke abbreviation (e.g., Control n)
  - typing assistance
  - list of possible commands in echo area
  - prompts in the echo area

# emacs--Help

---

- **written manual**
- **reference card from manual**
- **online tutorial**
- **online help system**
  - Control **h** help options
  - Control **h** Control **h** (possible help options, prompts you to type desired help option)
  - third Control **h** displays what the option means

# emacs--Online Help Options

---

- **cancel commands**
  - Control **g**
- **examples of options in help**
  - Control **h t** (tutorial), Control **h a *word*** (all commands containing the word), Control **h b** (command dispatch table), Control **h k *key*** (name and information about the command key), Control **h l** (list last 100 characters typed), Control **h i** (run program for browsing files, including complete emacs manual)

# emacs--Try This

---

- **open your file**
- **set the line length for 64 characters**
- **Control h b** to get the dispatch command table
- **back in your file, type Escape next-line**
  - what appears in the echo area?
- **q** to quit the help pages
- **Control h a print**
- **Escape x n**
  - What happens
- **Control q**

# emacs--Search Options

---

- **search for text strings**
  - Control **s** *string* (incremental search forward for string), Control **r** *string* (incremental search backward for string)
- **search and replace**
  - Escape **<** (go to beginning of buffer), Escape **%** *book* Return *epic* Return (replace, depending on next key): Spacebar (make change and advance to next occurrence); Delete (skip change and advance to next occurrence); Escape (exit query-replace); **!** (replace all remaining occurrences); **^** (back up to previous occurrence); Control **h** (display help)

# emacs--Defining Regions

---

- **selecting text**
  - point
    - beginning of region
    - where the cursor is
  - mark
    - end of region
    - marked by Control @ or Control Spacebar
  - region
    - between point and mark
- **how to select text**
  - move cursor to beginning of region
  - Control @
  - move cursor to end of region
  - Control x Control x--check location of mark; this keystroke combination exchanges point and mark
  - act on region (see next slide)

# emacs--Acting on Regions

---

- **commands**

- `upcase-region` (Control **x** Control **u**)
- `downcase-region` (Control **x** Control **l**)
- `append-to-file` *file* (append region to a file)
- `write-region` *file* (write region to a file)
- `kill-region` (Control **w**) (kill the region)
- `copy-region-as-kill` (Escape **w**) (copy region to kill buffer)
- `fill-region` (Escape **g**) (justify region)

# emacs--Try This

---

- **open your file**
- **search forward for a character (Control s)**
- **search backwards for a character (Control r)**
- **search for a word and replace it with another**
  - Escape %*word* Return *word2* Return
  - watch what happens
- **define a region**
  - Control @ (at beginning)
  - move cursor to character after last of desired region
  - Control x Control x
  - Escape w
  - Escape y
  - watch what happens

# emacs--Formatting

---

- `auto-fill-mode` (word wrap at right margin)
- `fill-region` (justify region)
- Escape `q` (justify paragraph at right margin)
- Escape `n` Control `x` `f` (set right margin at `n` characters)

# emacs--Multiple Windows

---

- **splitting the screen**
  - display two parts of the same file
  - display two different files
- **commands**
  - Control **x** 2 (divide current window into two windows vertically)
  - Control **x** 3 (divide current window into two windows horizontally)
  - Control **x** 1 (delete all windows but the current one)
  - Control **x** 0 (delete current window, redistribute space)
  - Control **x** o (switch to other window, cycle through all)
  - Escape Control **v** (page other window)
  - Control **x** ^ (increase current window by one line vertically)
  - Control **x** } (increase current window by one line horizontally)

# emacs--Multiple Windows

---

- **create multiple buffers**
  - Control **x** **b** *file1* (create new buffer or switch to buffer named file1)
  - Control **x** **k** *file1* (kill the buffer called file1)
  - Control **x** Control **f** *file2* (find file2, put it in a buffer, switch to it)
  - Control **x** Control **b** (list all buffers in separate window)
- **work with multiple buffers and multiple windows**
  - most convenient way to display two buffers in two separate windows
    - start **emacs** on a file (first buffer)
    - **Control x4f** (or **b** or **.**) filename (display second file in own window)

# emacs--File Management

---

- **commands for dealing with files**

- `emacs file` (create a file)
- Control **x** Control **s** (save changes)
- Escape **x** `write-file file` (usually used to change file name)
- Escape **x** `append-region-to-file file` (must mark region)
- Control **x** **i** `file` (insert a file into buffer at cursor)
- Control **x** Control **r** `file` (read a file, no editing permitted)
- Control **x** Control **v** `file` (visit and replace a file)

# Other Editor--Pico

---

- **pico**
  - comes with the mail system, pine
  - very simple
  - menu based
  - limited, but easy to use
  - for more power, use vi
  - not always available
  - ask systems administrator for access to it

# Files and Directories, Part II

---

- [Files and Directories--Basic Concepts](#)
- [Manipulating Files and Directories](#)
- [Wildcards/Metacharacters](#)
- [Directory Abbreviations](#)

# Files and Directories--Basic Concepts

---

- **pathname**
  - path through the directory system to the file
  - example
    - /usr/Workshop1/Subdirectory/file
  - tail (basename)
    - last part of pathname
  - head
    - everything except the tail
  - full pathname
    - shown when you type `pwd`
- **/ (slash)**
  - two meanings
    - very first one means root or top of the file system
    - all others separate directory or file names from one another

# Manipulating Files and Directories

---

- `rm -i -r file`
  - irreversible
- `cp -i file1 file2 or file(s) pathname`
  - irreversible; make sure you aren't using an existing file name for file2
- `mv -i file1 file2 or file(s) pathname`
  - irreversible
- `ln [none] file1 name2 or file(s) pathname`
- `mkdir [none] directory`
- `rmdir [none] directory`
  - directory must be empty if you use `rmdir`; delete files in the directory, then run `rmdir`
  - irreversible
- `cd [none] directory`
- `pwd [none] [none]`

# Files and Directories--Try This

---

- create four short files; run list
- remove one file; run list
- remove one file with the `-i` option--what happens
- run list
- copy one file to another existing file: `cp file1 file2`
- run list; open file2; what has happened
- move one file to another existing file; open the new file
- run list; what has happened
- create a directory; run list
- change to that directory; print your working directory; run list
- go back to your home directory

# Files and Directories--Try This, cont'd

---

- copy a file to the new directory

```
$ cp filex directory/filex
$ ls
$ cd directory
$ ls
$ pwd
$ cd
$ pwd
```

# Wildcards/Metacharacters

---

- save time and typing
- pattern searching

# Wildcards/Metacharacters

---

- **?**
  - match one character, and one character only
- **\***
  - match zero or more characters
- **[ ]**
  - match one character of the ones listed inside the brackets

- **examples**

```
ls yo?
```

```
ls yo??
```

```
ls ?yo?
```

```
ls ?yo
```

```
ls [abc]
```

```
ls *[abc]
```

```
ls [a-f]
```

```
ls *
```

```
ls ?yo*
```

```
ls yo*
```

```
ls *yo
```

```
ls [abs]*
```

```
ls [xyz]a[rst]
```

```
ls [A-F]
```

# Wildcards/Metacharacters--Try This

---

- `ls`
- `pwd`
- **create new files**
  - alabama            `cat > alabama`
  - alaska             `cat > alaska`
  - florida             `cat > florida`
  - massachusetts     `cat > massachusetts`
  - connecticut        `cat > connecticut`
- **try listing these with different wildcards and metacharacters**

```
ls al*
```

```
ls ?la
```

```
ls [a-f]*
```

```
ls [a-f]labama
```

```
ls m*
```

```
ls *a
```

```
ls ?la*
```

```
ls [b-z]labama
```

```
ls *[i]*
```

```
ls C*
```

# Directory Abbreviations

---

- **.** (one period)
  - current working directory
- **..** (two periods)
  - directory above the one in which you are working
- **~** (tilde)
  - your home directory
- **cd**
  - takes you to your home directory

# Permissions

---

- used to allow or restrict access to files and directories
- `chmod ugo±rwx file`    `chmod ugo±rwx directory`

```
$ ls -l
drwxr-xr-x ...
-rw-r--r-- ...
```

- d = directory
- columns 2, 3, 4: permissions for user (read, write, execute)
- columns 5, 6, 7: permissions for group
- columns 8, 9, 10: permissions for others
- change your home directory's permission by writing, e.g.,
  - `chmod g+rwx .`

# Permissions--Try This

---

- create a file called `filez`

```
$ ls -l
```

```
$ chmod g+rxw filez
```

```
$ ls -l
```

- make a directory

```
$ ls -l
```

- change to new directory

```
$ pwd
```

```
$ cd
```

```
$ chmod u-rwx directory
```

```
$ ls -l
```

```
$ cd directory
```

# Permissions--Try This, cont'd

---

- **what happens**

```
$ chmod u+rwx directory
```

```
$ ls -l
```

```
$ cd directory
```

```
$ pwd
```

# UNIX Shell

---

- [UNIX Shell--Basic Concepts](#)
- [Job Control](#)

# UNIX Shell--Basic Concepts

---

- **two functions**

- command interpreter between machine and user
- programming language
  - can string together basic commands to perform larger task
  - format

*command -option argument*

» separated by one space

- usually use the “pipe” (|) to combine commands
  - the pipe takes the output of one command and uses it as the input to another command

# Job Control

---

- **Job control**

Control **z**

- stop current job

**fg**

- resume stopped job

- **Multiple -l**

– list and label the jobs you've begun

```
[2]      Stopped      vi file1
[3] -   Stopped      vi file2
[4] +   Stopped      vi file3
[5]      Running      command
```

**fg %n**

- brings job number *n* to the foreground

**bg %n**

- sends job number *n* to the background

# Job Control, continued

---

- **Job control**

`$ ps`

PID	TT	STAT	TIME	COMMAND
431	A5	R	0:05	sort file1
432	A5	R	0:00	ps

- **Killing a job that is out of control**

`kill %I` or `kill pid`

– even more sure kill: `kill -9 %n`

# Selected References

---

- Lasser, Jon. *Think UNIX*. QUE Corporation
  - Levine, John R. and Margaret Levine Young. *UNIX for Dummies*. Dummies Press, a division of IDG Books Worldwide, Inc.
  - Martin, Don, et al. *UNIX Primer Plus*. Waite Group Press, A Division of Macmillan USA, Inc.
  - Ray, Deborah S. and Eric J. Ray. *Visual Quickstart Guide: UNIX*. Peachpit Press.
  - Reichard, Kevin and Eric Foster-Johnson. *UNIX in Plain English*. M&T Books.
  - Lamb, Linda and Arnold Robbins. *Learning the vi Editor*. O'Reilly
  - Pedersen, Jesper, et al. *SAMS Teach Yourself Emacs in 24 Hours*. Sams, A Division of Macmillan Computer Publishing.
-

# Exercises--Easy Commands

---

- type `fenger` and correct it to `finger`
- how do you determine all of the people logged on to the system
- how do you find the time of day
- how do you find the calendar for the year 1066
- on what day of the week was your birthday in 1995
- type a command and then kill the whole line
- find out what options you can use with the command `who`

# Exercises--Easy Commands: Answers

---

- use the backspace to delete characters back to the incorrect one, then retype
- type `who`
- type `date`
- `cal 1066`
- `cal month year`
- use **Control U** to delete an entire line
- type `man who` to get the whole online manual page for `who`

# Exercises--Files--Creating

---

- list your home directory's contents, using different options and combinations of options
- create a file using redirection containing the output of the command `ls -l`
- type the command `ls`; what happens
- use `cat` to create a new file called *filea*; use `cat` to create another file called *fileb*; use redirection to create a new file called *filec* containing the contents of both *filea* and *fileb*; look at the contents of *filec*
- use redirection to add the contents of *filea* to *filec*; look at the contents of *filec*; what has happened?

# Exercises--Files--Creating: Answers

---

- `ls`, `ls -a`, `ls -l`, `ls -al`
- `ls -l > filename`
- you get a “command not found” message; when you type something that is not a command, either nothing happens or you get a little error message
- `cat > filea` (add some short text)  
`cat > fileb` (add some short text)  
`cat filea fileb > filec`  
`cat filec` (has the contents of both files)
- `cat filea > filec`  
`cat filec` (now has the contents of only `filea`)

Note the difference between redirecting a file to a new file and redirecting a file to an existing file; the latter is an irreversible overwriting of the existing file

---

# Exercises--vi Editor

---

- **create a file called *letter*; set the word wrap margin to 15; type (with appropriate end of line insertions)**
  - Dear Author, Never have I read such an interesting book. Your writing inspires me. I hope you write an infinite number of books.  
Sincerely, Reader
  - save the letter but don't exit it
  - change the word Reader to your name
  - join two lines
  - insert some text
  - undo some typing
  - save and close the letter and run a spelling check
  - reopen the file and correct typos
  - yank a line and paste it elsewhere
  - delete a line and paste it elsewhere

# Exercises--vi Editor: Answers

---

- `vedit letter` or `vi letter`
- `:set wm=15`
- type the text
- `:w!`
- `J`
- insert desired text using `i`, `I`, `a`, `A`, `o`, `O`
- from the insert mode: Escape `u`
- `ZZ` at the shell prompt type `spell letter`
- `vedit letter` correct typos by moving cursor, using `x` to delete a character, and then entering insert mode with `i` or by using `r` to overwrite a character without entering insert mode

# Exercises--vi Editor: Answers, cont'd

---

- move cursor to line you want to yank

`yy`

move cursor to line where you want to place the yanked line

`p`

- move cursor to line you want to delete and paste elsewhere

`dd`

move cursor to line where you want to place the deleted line

`p`

# Exercises--emacs Editor

---

- **create a file with emacs**  
**set line length to 64**  
**type a letter**  
**save without exiting**  
**search for a string**  
**delete a line and put it elsewhere**  
**exit the file without saving changes**  
**reopen the file search for a word and replace it with another word**  
**practice search/replace with options**
- **look at the dispatch command table**
- **look at the tutorial and move through it**

# Exercises--emacs Editor, cont'd

---

- open your file  
define a region  
capitalize all the letters in that region

# Exercises--emacs Editor--Answers

---

- `emacs file` (to open a new file)  
Escape 64 Escape set fill-column (set line length)  
Control x Control s (save without exiting)  
Control s *string* (to look for some string; string could be a letter, a character, or more than one characters)  
Control k (move cursor) Control y (to delete a line and place it elsewhere)  
Control x Control c (to close the file without saving)  
`emacs file` (to reopen an existing file)  
Escape % *word1* Return *word2* Return (to search for word1 and replace it with word2)  
Control hb (to look at the dispatch command table)  
Control ht (to look at the tutorial)  
Control g (to cancel any command)
-

# Exercises--emacs Editor--Answers, ctd

---

- `emacs filename` (your existing file)  
Control @ (select the point)  
move cursor to end of region  
Control x Control x (indicate the mark)  
Control x Control u

# Exercises--after last section

---

- create files
- create directories
- watch what happens when you move files (`mv`), copy files (`cp`), remove files (`rm`)
- create subdirectories and move into them (`cd`), use `pwd` to show where you are; `cd` up one level
- use `rmdir` to delete a directory (directory must be empty)
- change permissions and run `ls -l` to see the differences
- run `ls -al` to look at the permissions on your home directory (the file indicated by `.-` a single dot)
- change permissions on your home directory to remove execute privileges from others (`chmod o-x`)
- PRACTICE, PRACTICE, PRACTICE
- TAKE THE NEXT UNIX WORKSHOP AT OSC