Introduction to UNIX
Basic Commands

CS2600 Systems Programming

Dr. Amar Raheja

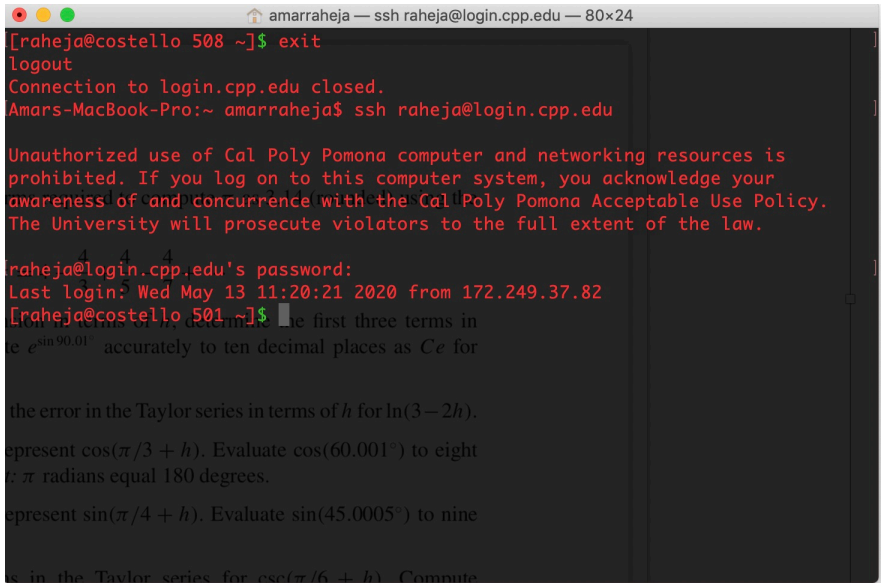
Lecture 3

Logging into UNIX machine

- If you have a Linux or Mac
  - Just open a terminal Window
- The Cal Poly Unix servers are:

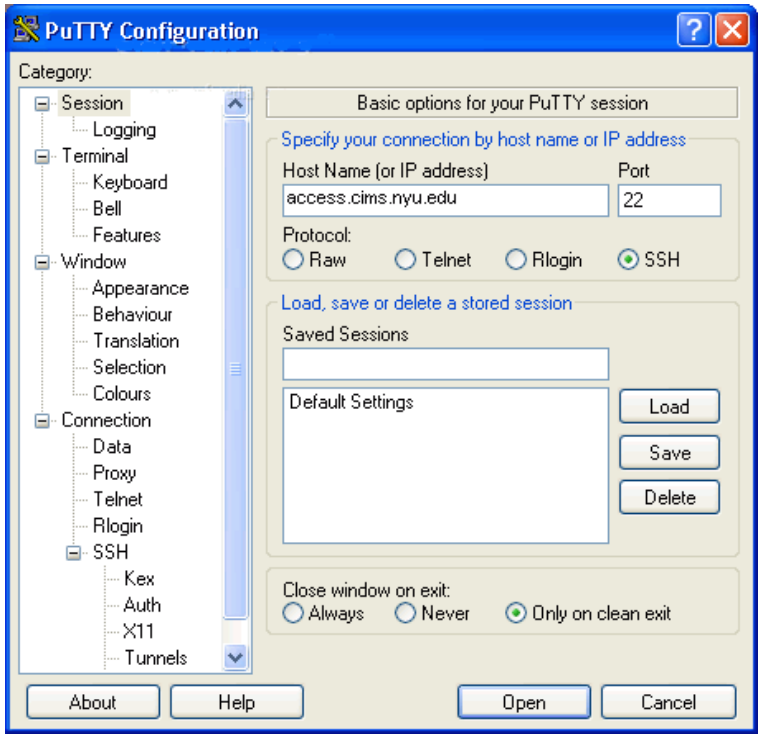
  login.cpp.edu

  files.cpp.edu (only for files)



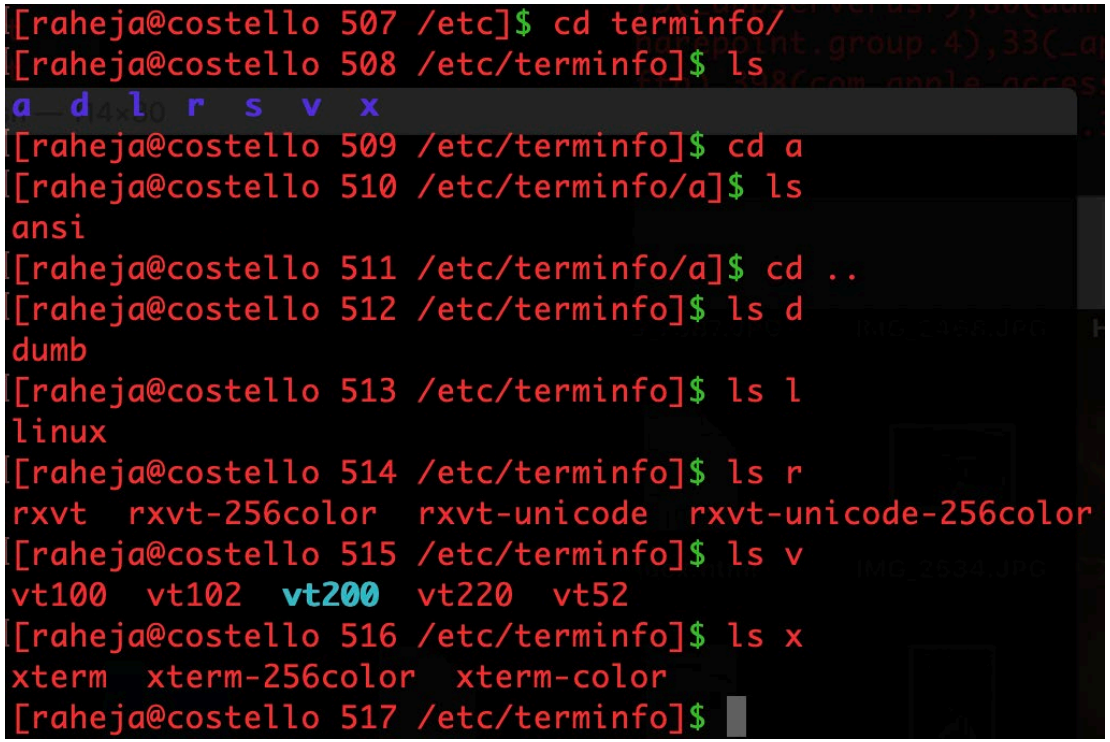
- Please try on your machines

- For Windows, please install and use it to ssh into Cal Poly Unix Server



Terminal Emulation (1 of 2)

- **Terminal emulation** is negotiated between your **client** machine and the **Unix server.**
- Your **virtual terminal ID** (e.g. **`pts/0`**) and **type** (e.g. vt100) are negotiated when you first connect and login.



To change the terminal type:

in bash:

```
export TERM=vt100
```

in bourne shell or ksh:

```
TERM=vt100
export TERM
```

in csh or tcsh:

```
setenv TERM vt100
```

Terminal Emulation (2 of 2)

`who am i` identifies your **username** and **terminal ID**.

```
$ who am i
raheja   pts/1       2020-05-27 21:54 (174.248.39.81)
```

`echo $TERM` identifies your login **terminal type**.

```
$ echo $TERM
xterm-256color
```

Special Control Keys

<CTRL>c	interrupt (stop program/command)
<CTRL>d	halt or EOF
<CTRL>g	bell
<CTRL>h	backspace
<CTRL>l	redraw screen
<CTRL>u	kill (erase) line
<CTRL>w	kill word
<CTRL>z	suspend
<CTRL>s	stop the screen from scrolling
<CTRL>q	continue scrolling

Shell and its Environment

- Shell is a program that runs upon login
  - The user interface to the operating system
- Functionality:
  - Execute other programs
  - Manage files
  - Manage processes

When you log in, you interactively use the shell:

- Command history
- Command line editing
- File expansion (tab completion)
- Command expansion
- Key bindings
- Spelling correction
- Job control

- Files help shape the environment upon login

`.profile`	Bourne/Korn shells
`.login`	C shell

The Unix Prompt

- After you log in, and the shell startup files have been run, the shell will display a prompt

  $

- Different shells and different systems have different **prompts**.
  - Two most common prompts are **$** and **#**. Also can be %
  - Your prompt can be changed using resource configuration files for the shells
- A prompt (plus a **cursor**) tells you that the system is ready for your commands.

Your Home Directory

`.`	Current **working** directory
`..`	Parent directory
`.login`	**login** script file (csh)
`.profile`	**login** script file (sh/ksh)
`.logout`	**logout** script file
`.plan`	**finger** resource file
`.cshrc`	**resource configuration** script file for C shell
`.bashrc`	**resource configuration** script file for Bourne Again shell
`.exrc`	**resource configuration** script file for vi

Standard Command Format (1 of 2)

Format: **`command [options] <arguments>`**

- stuff in brackets is optional
- boldface words are literals (must be typed as is)
- **<>** enclosed words are args (replace appropriately)

- Commands are case sensitive (mostly lowercase)
- Spaces must be inserted between commands, options, arguments
- Examples:

ls –al or ls –a –l or ls –F /dev

date

echo $TERM

Standard Command Format (2 of 2)

- **Options** (also called **flags**) modify how the command works (command behavior)
  - single letters prefixed with a dash “–”
  - combined or separated (e.g., –al = –a –l)
  - come before arguments

- **Arguments** define the command scope
  - Optional for some commands, mandatory for others
  - Some commands assume a default argument if none is supplied
  - Usually files or directories

Getting Help

- Check the **manual pages**!
  - For shell command, system programs, and library functions or anything else
- Format: man <command>

  man –k <keywords>
- Man(ual) page format

  Name

  Synopsis

  Description (options, defaults, detail desc., examples)

  Files

  See Also

  Bugs

man Examples

$ man man

Displays help on the **man** command

$ man who

Displays help on the **who** command

$ man -k mail

Checks all man pages for keyword “mail”

man Output Example

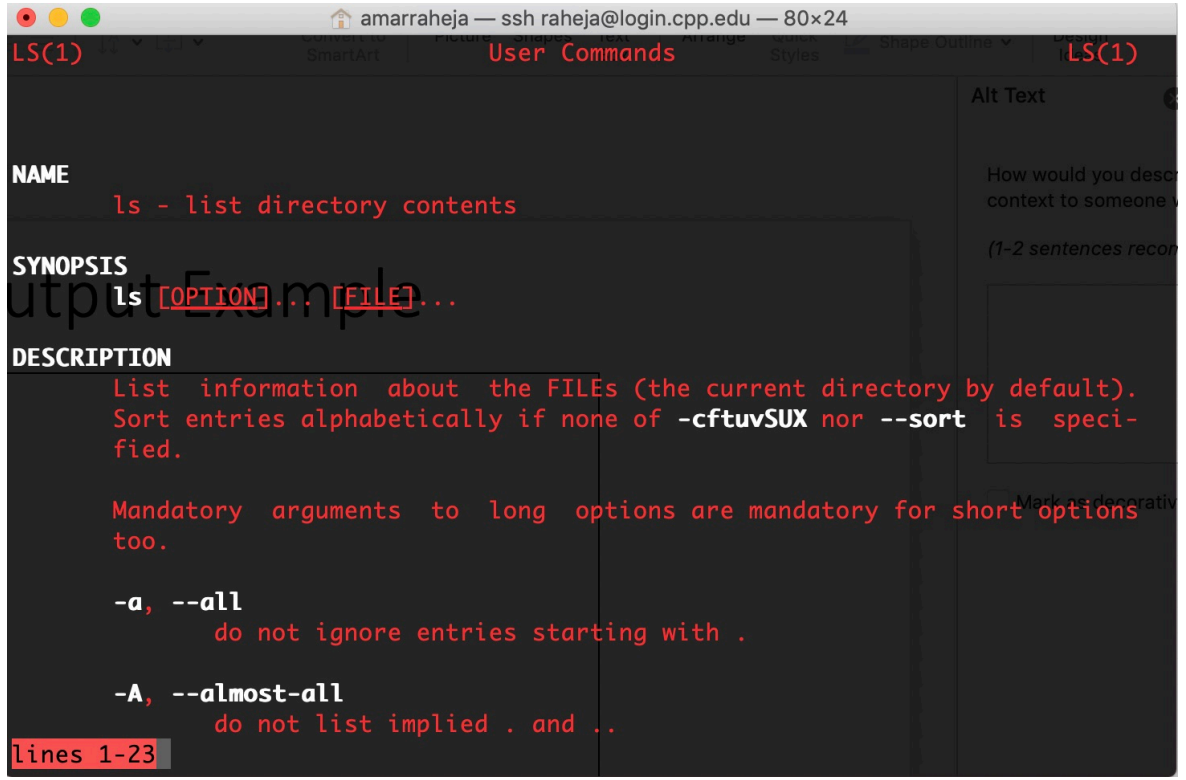
```
$ man ls
Reformatting page.  Wait... done

User Commands                                   ls(1)


NAME
   ls - list contents of directory


SYNOPSIS
   /usr/bin/ls [ -aAbcCdfFgilLmnopqrRstux1 ] [ file...  ]
   /usr/xpg4/bin/ls [ -aAbcCdfFgilLmnopqrRstux1 ] [ file...  ]


--More--(11%)
```



spacebar - move forward one page b – move back one page h – more commands q – quit

The `stty` Command

`stty` **set terminal type** options

change and print terminal line settings.

Basically, this command shows or changes terminal characteristics.

`stty -a` list **all** terminal settings

`stty erase ^h`

- **erase key is now `<CTRL>h`** **(and now the Backspace Key works!)**

Basic Commands to get started

date - Print the date and time

```
$ date
Wed 13 May 2020 05:40:30 PM PDT
$
```

echo - Display command line input to screen

```
$ echo Hi, I am Amar Raheja, your professor!
Hi, I am Amar Raheja, your Professor!
$
```

Commands to Manipulate Files or Directories

- *ls* lists files in a directory (names, not the contents of files)
- *cat, head, tail, page, more* display contents of files
- *rm* removes files (and directories)
- *cp* copies files (and directories)
- *mv* moves (renames) files (and directories)
- *cd* changes directories
- mkdir make empty directories
- *rmdir* remove empty directory
- *pwd* display name of present working directory

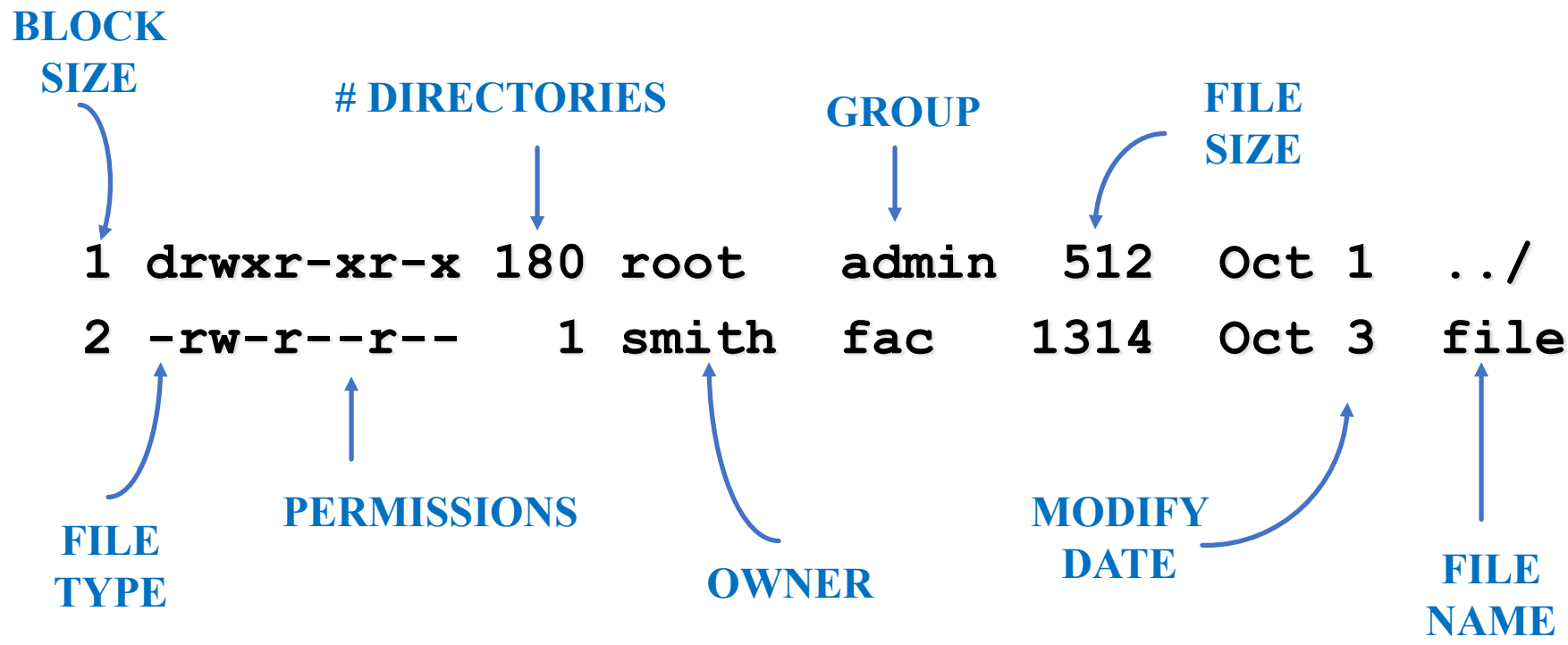
List Files in a Directory

Format: ls [-alRF…] <file-list>

- **`-a`** list all files including the dot files
- **`-l`** long format (show file type, permissions, #links, owner, etc)
- **`-R`** recursive list subdirectories
- **`-F`** list directories with file type (trailing / *)

Listing Files in a Directory

$ ls –al



Viewing Files

cat	concatenate and print to screen (ctrl-s and ctrl-q to stop/start)
head -x	display first x lines of file
tail -x	display last x lines of file (both default to 10 lines)
page	page file to the screen
more	display part of file to screen

Example: cat, head, tail

```
$ cat letter
Mr. Jones,
It is getting late. Please order some pizza and stop
by my office. We will tidy up a few more things before
calling it a night.
Thanks!
Ben
$ head -2 letter
Mr. Jones,
It is getting late. Please order some pizza and stop
$ tail -1 letter
Ben
```

Copying Files

Format:

```
cp [-ir…] file1 file2
cp [-ir…] file-list directory
cp [-ir…] directory directory
```

- i for interactive. Prompt whenever a file will be overwritten.
- r for recursive. copy a whole directory tree.

cp Examples

```
$ ls
letter1  secret
$ cp letter1 letter2
$ ls -F
letter1  letter2  secret/
$ cp letter1 letter2 secret
$ ls secret
letter1  letter2
```

Moving / Renaming Files

Format:

`mv [-i…] file1 file2`

- Renames file1 to file2

`mv [-i…] file-list directory`

- Moves files from current location to new directory

`mv [-i…] directory directory`

- Renames a directory

mv Examples

```
$ ls
letter      memo        saved
$ mv memo memo1
$ ls -F
letter      memo1 saved/
$ mv saved trash
$ ls -F
letter      memo1 trash/
```

Deleting Files

Format:

`rm file-list`

- Deletes files

`rm -r directory`

- Deletes directory and all files and directories within it
- Use with CAUTION!

`rm -i file`

- Deletes file with inquiry
- This is the preferred way to delete to prevent accidental deletion as file cannot be recovered once deleted

rm Examples

```
$ ls -F
letter1   letter2   secret/
$ rm -i letter1
rm: remove regular file ‘letter1’? y
$ ls -F
letter2   secret/
$ ls -F secret
memo      morestuff/
$ rm -r secret
$ ls
letter2
```

Finding User Information

- **who** Who is logged on, when & where

```
$ who
denni221   pts/3   May  5 18:39 (164.47.158.163)
small000   pts/4   May  5 18:51 (mackey.rbe36-16.den.pcisys.net)
```

- **finger** A bit more login information but might be disabled

```
$ finger
Login    Name                    TTY         Idle When            Where
denni221 Student - Victoria N    pts/3          1 Mon 18:39  164.47.158.163
small000 Faculty - Pam Small     pts/4           Mon 18:51  mackey.den.net
```

Communicating with others

- write - one way messaging

```
$ write user
```

- talk - interactive messaging

```
$ talk user
```

- e-mail programs

  `mail` - simple and old email program

  `mailx` - newer, improved email

write Example

- **`write`** Send one way message to another user

```
$ whoami
smith321
$ write jones456
Bill, you've been idle for a
long time! What are you doing?
[hit CTRL-D to end write message]
$
----------------------------------------------------
$ whoami
jones456
  Message from smith321 [Fri Mar 29 20:18:47]
Bill, you've been idle for a
long time! What are you doing?
<EOT>
$
```

talk Example (screen 1)

- **`talk`** Instant messaging for UNIX

```
$ whoami
smith321
$ talk jones456
[Waiting for your party to respond]
[Connection established]
Hi Bill, what's up?


+----------------------------------------+
Hi! I'm a little busy right now.
Is it okay if I call you back latter?
```

talk Example (screen 2)

```
$ whoami
jones456
$
Message from Talk_Daemon at 20:41 ...
talk: connection requested by smith321
talk: respond with:  talk smith321
$ talk smith321
[Waiting for your party to respond]
[Connection established]
Hi! I'm a little busy right now.
Is it okay if I call you back latter?

+----------------------------------------+
Hi Bill, what's up?
```

Process Subsystem utilities

- **ps** monitors status of processes
- **kill** send a signal to a pid
- **wait** parent process wait for one of its children to terminate
- **nohup** makes a command immune to the hangup and terminate signal
- **sleep** sleep in seconds
- **nice** run processes at low priority

Processes

- ps tells us about all processes being run by the current user

```
$ ps
  PID  TTY          TIME    CMD
68468 ttys000    0:00.05  -bash
```

- Please try these commands

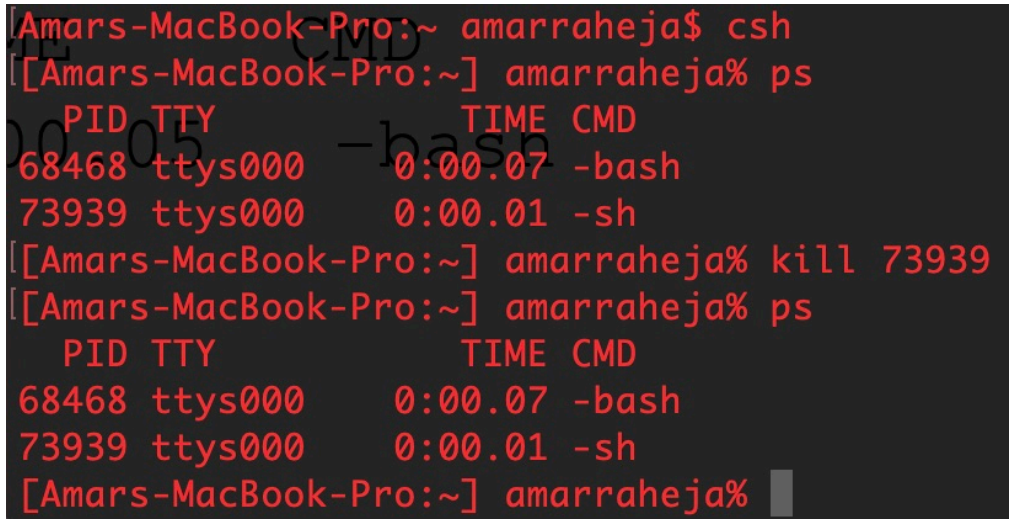
```
$ ps -u <username>
$ ps -u root
$ ps -f
 UID   PID    PPID  C    STIME   TTY    TIME     CMD
 504 68468 68466    0  5:28PM ttys000 0:00.07  -bash
```

kill and sleep Example

$ kill

- The kill utility sends a signal to the processes specified by the pid operands
- Some of the more commonly used signals:

1	HUP (hang up)
2	INT (interrupt)
3	QUIT (quit)
6	ABRT (abort)
9	KILL (non-catchable, non-ignorable kill)
14	ALRM (alarm clock)
15	TERM (software termination signal)



$ sleep 5

Puts the terminal to sleep for 5 seconds