CS395T: Introduction to Scientific and Technical Computing

Instructors:

Dr. Karl W. Schulz, Research Associate, TACC
Dr. Victor Eijkhout, Research Scientist, TACC

Outline

- Finish Batch System
- Unix Introduction

LSF: Job Script Submission

- When submitting jobs to LSF using a job script, a redirection is required for bsub to read the commands. Consider the following script:

```
lslogin1> cat job.script
#!/bin/csh
#BSUB -n 32
#BSUB -J hello
#BSUB -o %J.out
#BSUB -e %J.err
#BSUB -q normal
#BSUB -W 0:15
echo "Master Host = "`hostname`
echo "LSF_SUBMIT_DIR: $LS_SUBCWD“
echo "PWD_DIR: "`pwd`

ibrun ./hello
```

- To submit the job:

```
lslogin1% bsub < job
```

Re-direction is required!

LSF: Interactive Execution

- Several ways to run interactively
  - Submit entire command to bsub directly:

```
> bsub –q development -I -n 2 -W 0:15 ibrun ./hello

Your job is being routed to the development queue
Job <11822> is submitted to queue <development>.
<<Waiting for dispatch ...>>
<<Starting on compute-1-0>>
 Hello, world!
   --> Process #   0 of   2 is alive. ->compute-1-0
   --> Process #   1 of   2 is alive. ->compute-1-0
```

  - Submit using normal job script and include additional `-I` directive:

```
> bsub -I < job.script
```

Batch Script Suggestions

- Echo issuing commands
  - (“set -x” and “set echo” for ksh and csh).
- Avoid absolute pathnames
  - Use relative path names or environment variables ($HOME, $WORK)
- Abort job when a critical command fails.
- Print environment
  - Include the "env" command if your batch job doesn't execute the same as in an interactive execution.
- Use “./” prefix for executing commands in the current directory
  - The dot means to look for commands in the present working directory. Not all systems include "." in your $PATH variable. (usage: ./a.out).
- Track your CPU time

LSF Job Monitoring (*showq* utility)

```
lslogin1% showq
ACTIVE JOBS--------------------
JOBID     JOBNAME    USERNAME      STATE  PROC  REMAINING       STARTTIME

11318  1024_90_96x6    vmcalo    Running    64   18:09:19  Fri Jan  9 10:43:53
11352           naf   phaa406    Running    16   17:51:15  Fri Jan  9 10:25:49
11357           24N   phaa406    Running    16   18:19:12  Fri Jan  9 10:53:46
 23 Active jobs     504 of 556 Processors Active (90.65%)

IDLE JOBS----------------------
JOBID     JOBNAME    USERNAME      STATE  PROC    WCLIMIT       QUEUETIME

11169        poroe8      xgai       Idle   128   10:00:00  Thu Jan  8 10:17:06
11645   meshconv019    bbarth       Idle    16   24:00:00  Fri Jan  9 16:24:18
 3 Idle jobs

BLOCKED JOBS------------------
JOBID     JOBNAME    USERNAME      STATE  PROC    WCLIMIT       QUEUETIME

11319  1024_90_96x6    vmcalo   Deferred    64   24:00:00  Thu Jan  8 18:09:11
11320  1024_90_96x6    vmcalo   Deferred    64   24:00:00  Thu Jan  8 18:09:11
 17 Blocked jobs

Total Jobs: 43    Active Jobs: 23    Idle Jobs: 3     Blocked Jobs: 17
```

LSF Job Monitoring (*bjobs* command)

```
lslogin1% bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
11635   bbarth  RUN   normal     lonestar    2*compute-8 *shconv009 Jan  9 16:24
                                             2*compute-9-22
                                             2*compute-3-25
                                             2*compute-8-30
                                             2*compute-1-27
                                             2*compute-4-2
                                             2*compute-3-9
                                             2*compute-6-13
11640   bbarth  RUN   normal     lonestar    2*compute-3 *shconv014 Jan  9 16:24
                                             2*compute-6-2
                                             2*compute-6-5
                                             2*compute-3-12
                                             2*compute-4-27
                                             2*compute-7-28
                                             2*compute-3-5
                                             2*compute-7-5
11657   bbarth  PEND  normal     lonestar                *shconv028 Jan  9 16:38
11658   bbarth  PEND  normal     lonestar                *shconv029 Jan  9 16:38
11662   bbarth  PEND  normal     lonestar                *shconv033 Jan  9 16:38
11663   bbarth  PEND  normal     lonestar                *shconv034 Jan  9 16:38
11667   bbarth  PEND  normal     lonestar                *shconv038 Jan  9 16:38
11668   bbarth  PEND  normal     lonestar                *shconv039 Jan  9 16:38
```

Note: Use "`bjobs -u all`" to see jobs from all users.

LSF Job Monitoring (*lsuser* utility)

```
lslogin1$ lsuser -u vap
JOBID   QUEUE    USER            NAME         PROCS    SUBMITTED
547741  normal      vap       vap_hd_sh_p96    14      Tue Jun  7 10:37:01
    2005

HOST            R15s    R1m     R15m    PAGES           MEM     SWAP    TEMP
compute-11-11     2.0     2.0     1.4     4.9P/s        1840M   2038M
    24320M
compute-8-3       2.0     2.0     2.0     1.9P/s        1839M   2041M
    23712M
compute-7-23      2.0     2.0     1.9     2.3P/s        1838M   2038M
    24752M
compute-3-19      2.0     2.0     2.0     2.6P/s        1847M   2041M
    23216M
compute-14-19     2.0     2.0     2.0     2.1P/s        1851M   2040M
    24752M
compute-3-21      2.0     2.0     1.7     2.0P/s        1845M   2038M
    24432M
compute-13-11     2.0     2.0     1.5     1.8P/s        1841M   2040M
    24752M
```

LSF Job Manipulation/Monitoring

- To kill a running or queued job (takes ~30 seconds to complete):

  ```
  bkill <jobID>
  bkill -r <jobID>
  ```

  (Use when bkill alone won't delete the job)

- To suspend a queued job:

  ```
  bstop <jobId>
  ```

- To resume a suspended job:

  ```
  bresume <jobID>
  ```

- To see more information on why a job is pending:

  ```
  bjobs –p <jobID>
  ```

- To see a historical summary of a job:

  ```
  bhist <jobID>
  ```

```
lslogin1> bhist 11821
Summary of time in seconds spent in various states:
JOBID   USER    JOB_NAME  PEND    PSUSP   RUN     USUSP   SSUSP   UNKWN   TOTAL
11821   karl    hello     131     0       127     0       0       0       258
```

Unix Outline

- What is “Unix” anyway?
  - historical background
  - major flavors of Unix
- Basic Unix concepts
  - user accounts
  - file system overview
  - how to get help
  - interacting with a login environment

Unix in Practice

- Q: Before we begin, does anyone have a feel for how many machines in the June 2006 Top500 list ran variants of the Unix operating System?

  What about Windows?

- A: 94.8% are UNIX-like

Operating System	Number of Systems	Percentage
Linux	367	73.40%
Unix	98	19.60%
Mac OS	5	1.00%
BSD Based	4	0.80%
Mixed	24	4.80%
Windows	2	0.40%

Unix Background

- Q: How old is Unix (5, 10, 20 years, or greater)?
- A: > 35 Years
  - Unix originally dates back to 1969 with a group at Bell Laboratories
  - The original Unix operating system was written in assembler
  - In 1973 Thompson and Ritchie finally succeeded in rewriting Unix in their new language. This was quite an audacious move; at the time, system programming was done in assembler in order to extract maximum performance from the hardware, and the very concept of a *portable* operating system was barely a gleam in anyone's eye
  - First Unix installations in 1972 had 3 users and a 500KB disk

DEC PDP-11, 1972

What is UNIX?

- UNIX is a multi user, preemptive, multitasking operating system which provides a number of facilities:
  - management of hardware resources
  - directories and file systems
  - loading / execution / suspension of programs

- What does UNIX stand for?
  - Nothing actually - It is a "play on words" of an older multiuser time-sharing OS known as Multics

- There are many flavors of UNIX:
  - Solaris (Sun)
  - AIX (IBM)
  - Tru64 (Compaq)
  - IRIX (SGI)
  - SysV (from AT&T)
  - BSD (from Berkeley)
  - Linux (its not UNIX, but it’s close enough)

What is Linux?

- Linux is a clone of the Unix operating system written from scratch by Linus Torvalds with assistance from developers around the globe (technically speaking, Linux is not UNIX)
- Torvalds uploaded the first version of Linux in September 1991
- Only about 2% of the current Linux kernel is written by Torvalds himself but he remains the ultimate authority on what new code is incorporated into the Linux kernel.
- Developed under the GNU General Public License , the source code for Linux is freely available
- Download latest kernels from www.kernel.org
- A large number of Linux-based distributions exist (for free or purchase):
  - RedHat, Fedora, CentOS
  - SUSE
  - Debian
  - Gentoo
  - Slackware
  - Ubuntu
  - Mandrake

Why use UNIX?

- **Performance:** as we've seen, supercomputers generally run UNIX; rich-multi user environment

- **Functionality:** a number of community driven scientific applications and libraries are developed under UNIX (molecular dynamics, linear algebra, fast-fourier transforms, etc).

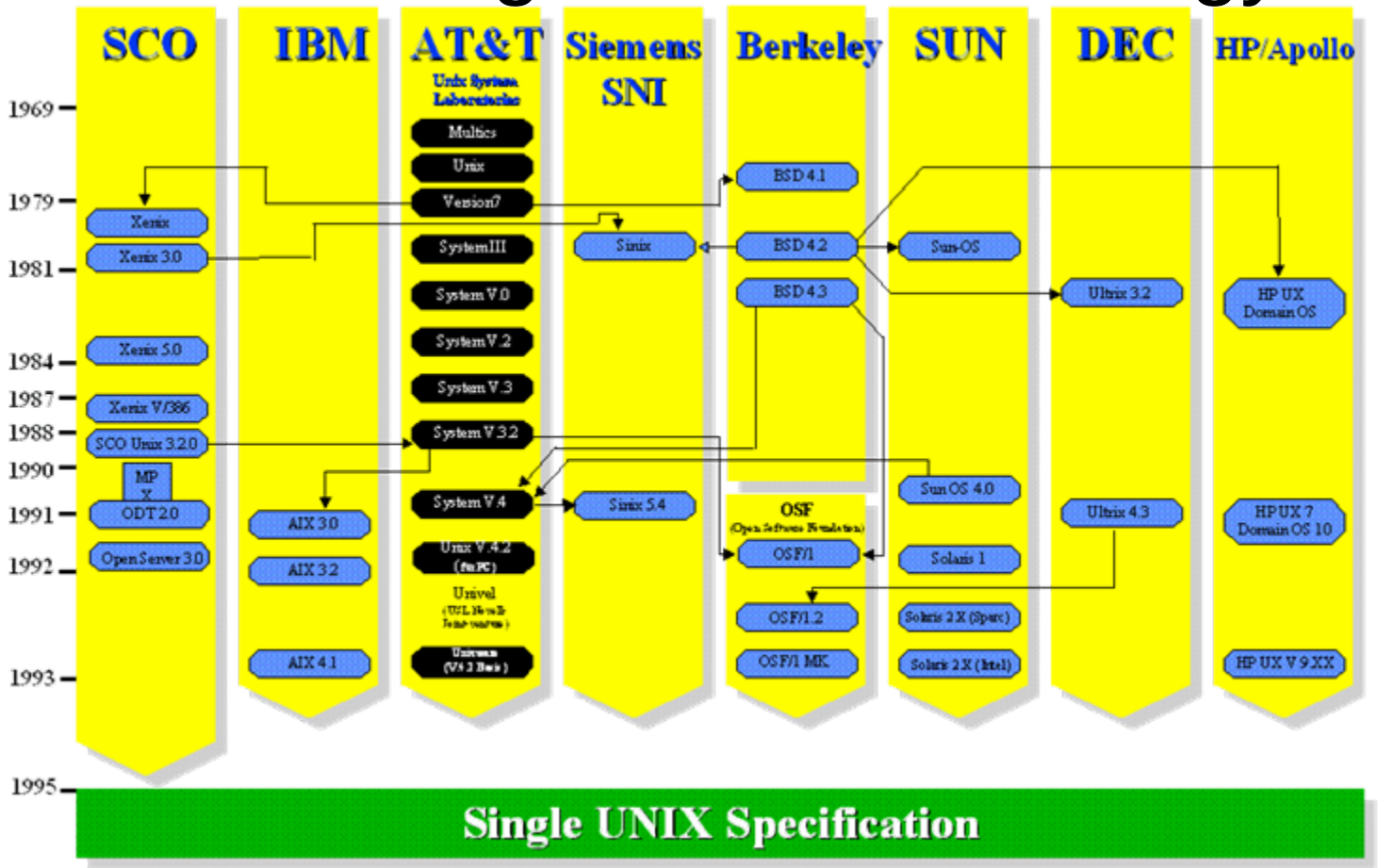
- **Flexibility/Portability:** UNIX lets you build your own applications and there is a wide array of support tools (compilers, scientific libraries, debuggers, network monitoring, etc.)

Some Key People

Ken Thompson and Dennis Ritchie
Your new heroes.

????
Linus Torvalds

Unix Background: Chronology

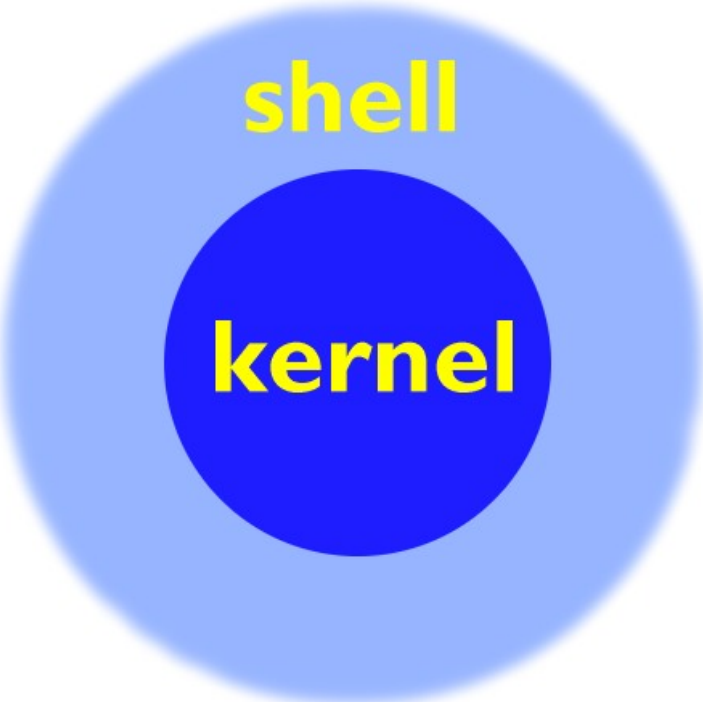


The Single UNIX Specification is the collective name of a family of standards for computer operating systems to qualify for the name "Unix“ (eg. HP-UX, IBM AIX, SGI IRIX, Sun Solaris).

Source: The Open Group, www.unix.org

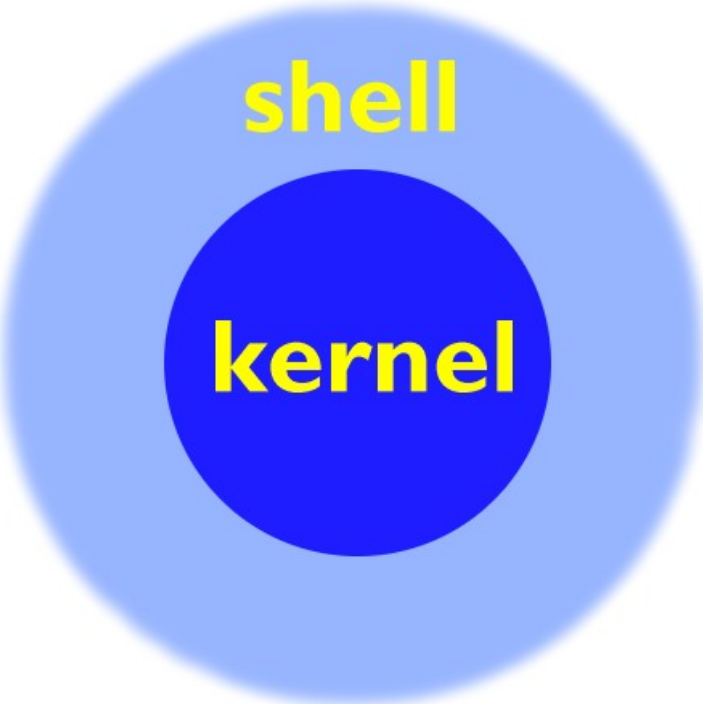
How does UNIX work?

- UNIX has a kernel and one or more shells
- The kernel is the core of the OS; it receives tasks from the shell and performs them
- The shell is the interface with which the user interacts



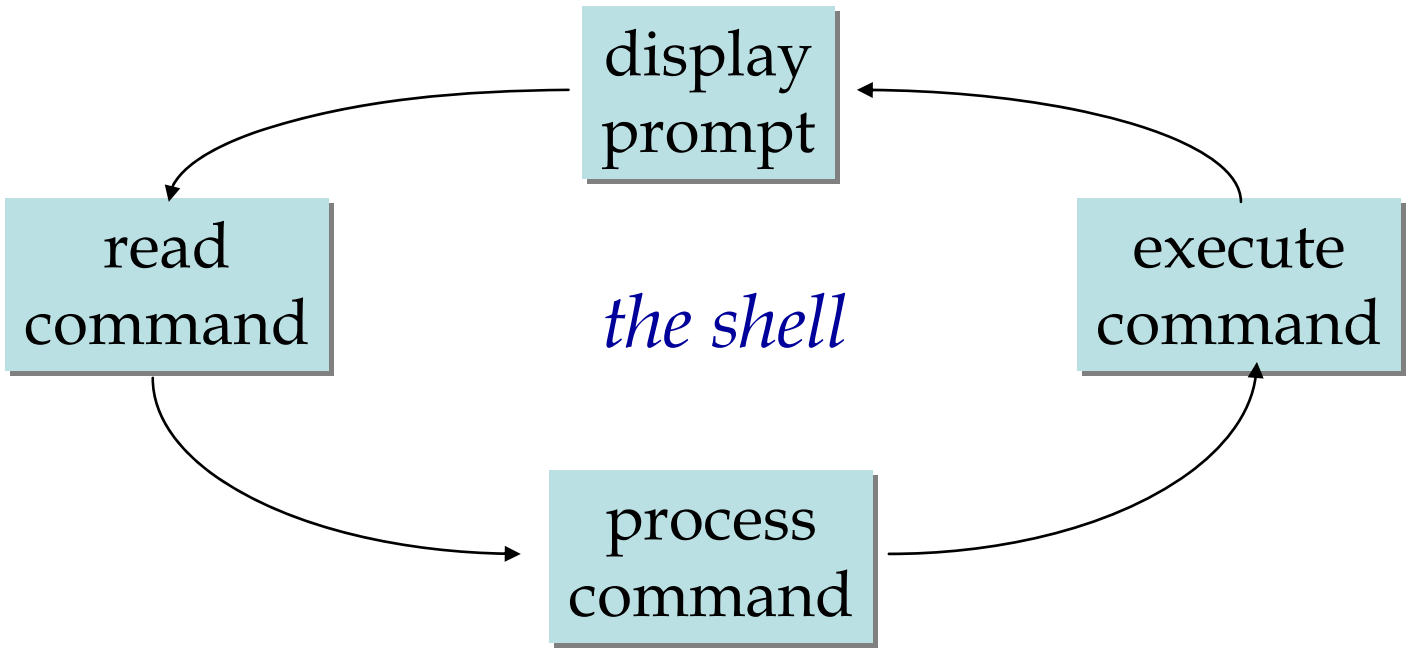
How does UNIX work?

- Everything in UNIX is either a *file* or a *process*
- A process is an executing program identified by a unique PID (process identifier). Processes may be short in duration or run indefinitely
- A file is a collection of data. They are created by users using text editors, running compilers, etc
- The UNIX kernel is responsible for organizing processes and interacting with files: it allocates time and memory to each processes and handles the filesystem and communications in response to system calls



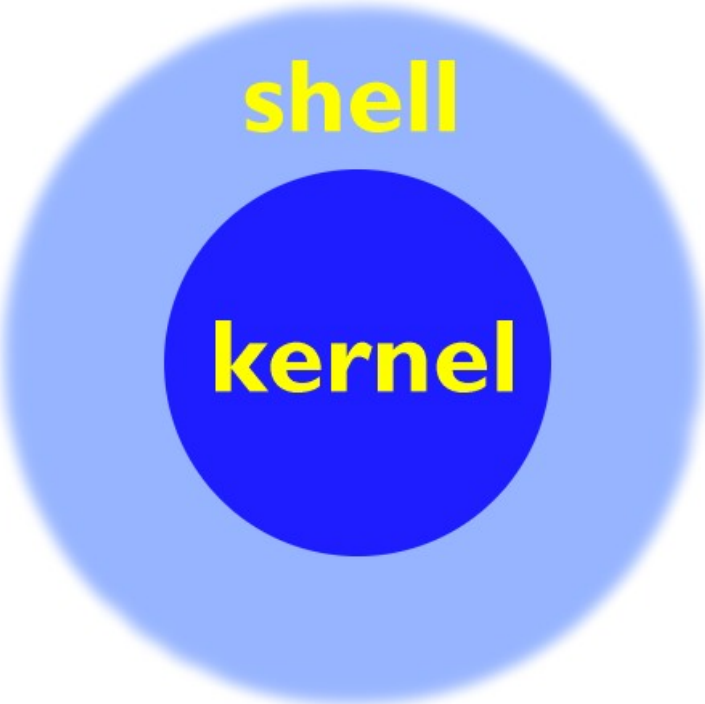
What does the Shell Do?

- The UNIX user interface is called the *shell*.
- The shell tends to do 4 jobs repeatedly:



An Example

- Example: Suppose a user wants to remove a particular file:
  - User has a command-line prompt (the shell is waiting for instructions)
  - User types a command requesting the file removal (eg. rm myfile) in the shell
  - The shell searches the filesystem for the file containing the remove program (rm)
  - A new process is forked from the shell to run the command with an instruction to remove myfile
  - The process requests that the kernel, through system calls, delete the reference to myfile in the filesystem
  - When the rm process is complete, the shell then returns to the UNIX prompt indicating that it is waiting for further commands
  - The process ID (PID) originally assigned to the rm command is no longer active



Unix Interaction

- The user interacts with UNIX via a shell
- The shell can be graphical (X-Windows) or text-based (command-line) shells like tcsh and bash
- To remotely access a shell session on TACC production resources, use ssh (secure shell)
- ssh is a secure replacement for telnet

X-Windows and Unix

- X-Windows is the standard graphical layer for UNIX systems
- Most graphical interfaces for UNIX are actually built on top of X-Windows
- Fundamental command-line application in X-windows is an *xterm*

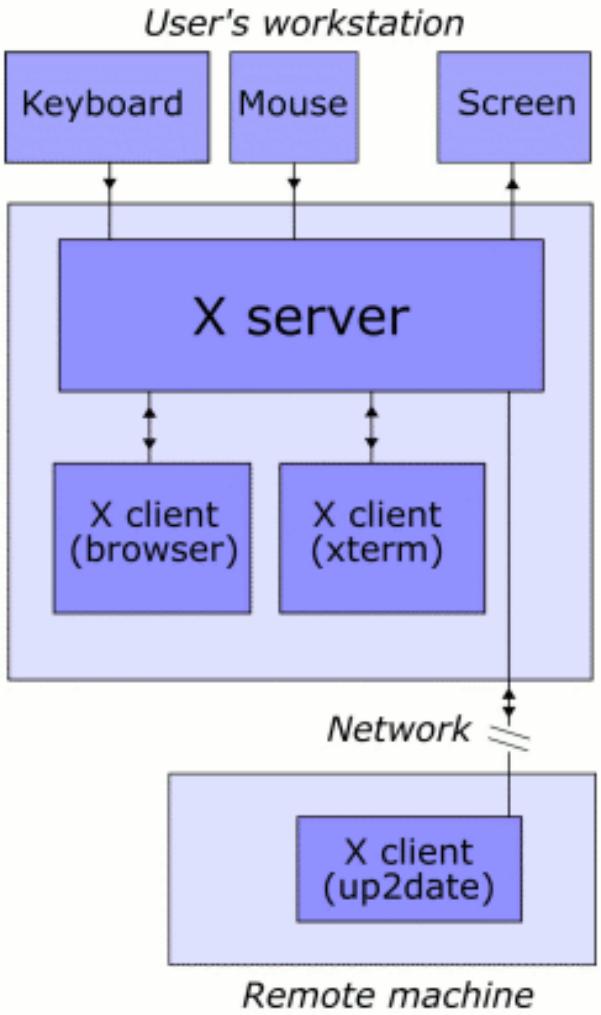
```
[/home]$ ls
vidarlo
[/home]$ cd ..
[/]$ cd etc
[/etc]$ ls
0.0.10.in-addr.arpa  csh.cshrc            gshadow-          logrotate.d       odbcinst.ini            rmt
adduser.conf         csh.login            gtk               lynx.cfg          openoffice              rpc
adjtime              csh.logout           host.conf         magic             opt                     screenrc
aliases              db.cache             hostname          mailcap           pam.conf                securetty
alternatives         debconf.conf         hosts             mailcap.order     pam.d                   security
apm                  debian_version       hosts.allow       mailname          passwd                  services
apt                  default              hosts.deny        mail.rc           passwd-                 shadow
asterisk             defoma               hotplug           manpath.config    perl                    shadow-
at.deny              deluser.conf         hotplug.d         mdadm             ppp                     shells
bakipkungfu          dhclient.conf        identd.conf       mediaprm          printcap                skel
bash.bashrc          dhclient-script      identd.key        mime.types        profile                 squid
bash_completion      dictionaries-common  inetd.conf        mkinitrd          protocols               ssh
bash_completion.d    discover.conf        init.d            modprobe.d        python2.3               sudoers
bind                 discover.conf-2.6    inittab           modules           raidtab                 sysctl.conf
blkid.tab            discover.d           inputrc           modules.conf      rc0.d                   syslog.conf
blkid.tab.old        dpkg                 ipkungfu          modules.conf.old  rc1.d                   terminfo
calendar             emacs                issue             modutils          rc2.d                   timezone
chatscripts          emacs21              issue.net         motd              rc3.d                   ucf.conf
chkrootkit.conf      email-addresses      kernel-img.conf   mtab              rc4.d                   updatedb.conf
complete.tcsh        environment          ldap              mtools.conf       rc5.d                   vidarlo.net.hosts
console              exim4                ld.so.cache       Muttrc            rc6.d                   w3m
console-tools        fdmount.conf         ld.so.conf        mysql             rc.d                    wgetrc
cron.d               fonts                locale.alias      nanorc            rcS.d                   #wvdial.conf#
cron.daily           fstab                locale.gen        network           reportbug.conf          wvdial.conf
cron.hourly          groff                localtime         networks          resolvconf              wvdial.conf~
cron.monthly         group                logcheck          nsswitch.conf     resolv.conf             X11
crontab              group-               login.defs        ODBCDataSources   resolv.conf~            xpilot
cron.weekly          gshadow              logrotate.conf    odbc.ini          resolv.conf.pppd-backup
[/etc]$
```

- A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell)

TACC

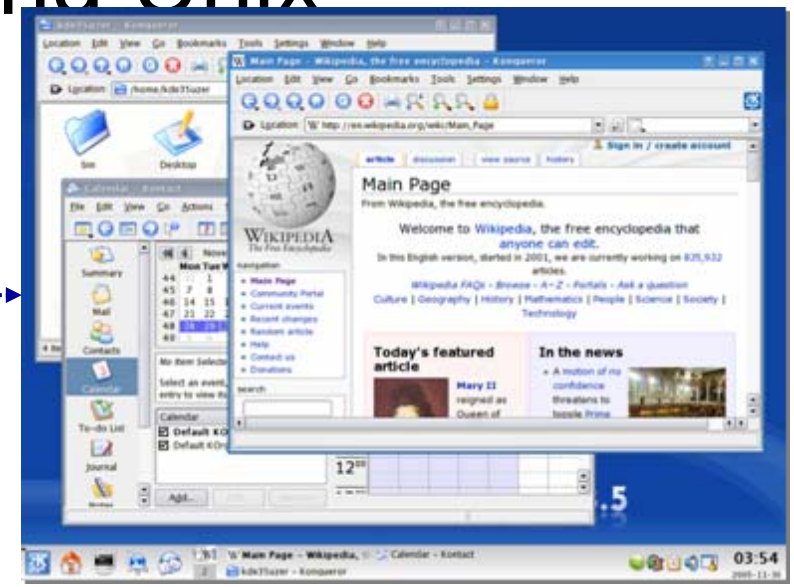
X-Windows

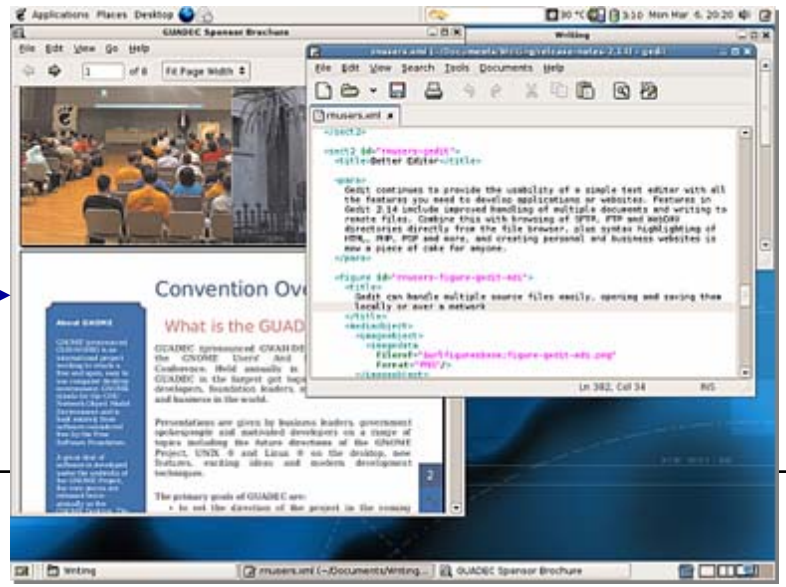
- The original idea of X emerged at MIT in 1984
- It provides a standard toolkit and protocol to build graphical user interfaces (GUI) on Unix, or Unix-like operating systems
- X supports remote connectivity
- The computer where application programs (the *client* applications) run can differ from the user's local machine (the display *server*).
- X's usage of the terms "client" and "server" reverses what people often expect, in that "server" refers to the user's local display ("display server") rather than to a remote machine.



X-Windows and Unix

- Several nice desktop environments exist for Linux
  - KDE
  - Gnome
- Cygwin for Windows also includes an Xserver and xterm client
- XFree86 is a freely redistributable open-source implementation of the X Window System (www.xfree86.org)





Accounts and the Unix File System

Unix Accounts

- To access a Unix system you need to have an *account*
- Unix account includes:
  - username and password
  - userid and groupid
  - home directory
    - a place to keep all your snazzy files
    - may be quota'd, meaning that the system imposes a limit on how much data you can have
  - a default shell preference

Unix Accounts

- A username is (typically) a sequence of alphanumeric characters of length no more than 8:
  - *eg. koomie* or *istc00*, *istc01, …*
- The username is the primary identifying attribute of your account
- the name of your home directory is usually related to your username:
  - *eg. /home/utexas/istc/istc00*

Unix Accounts

- A password is a secret string that only the user knows (not even the system knows it)
- When you enter your password the system encrypts it and compares to a stored string
- passwords are (usually) no more than 8 characters long.
- It's a good idea to include numbers and/or special characters (don't use an english word, as this is easy to crack)

Unix Accounts

- A *userid* is a number (an integer) that identifies a Unix account. Each userid must be unique
- In Unix-speak, userid's are known as *UID's*
- Why does Unix implement UID's? It's easier (and more efficient) for the system to use a number than a string like the username
- You don't necessarily need to know your userid

Unix Accounts

- Unix includes the notion of a "group" of users
- A Unix group can share files and active processes
- Each account is assigned a "primary" group
- The *groupid* is a number that corresponds to this primary group
- In Unix-speak, groupid's are knows as *GID's*
- A single account can belong to many groups (but has only one primary group)

Files and File Names

- A file is a basic unit of storage (usually storage on a disk)
- Every file has a name
- Unix file names can contain any characters (although some make it difficult to access the file)
- Unix file names can be long!
  - how long depends on your specific flavor of Unix

File Contents

- Each file can hold some raw data
- Unix does not impose any structure on files
  - files can hold any sequence of bytes
  - it is up to the application or user to interpret the files correctly
- Many programs interpret the contents of a file as having some special structure
  - text file, sequence of integers, database records, etc.
  - in scientific computing, we often use binary files for efficiency in storage and data access
    - Fortran unformatted files
    - Scientific data formats like NetCDF or HDF have specific formats and provide APIs for reading and writing
    - Portability is an issue with some formats (*little endian vs. big endian)*

Directories

- A directory is a special kind of file - Unix uses a directory to hold information about other files
- We often think of a directory as a container that holds other files (or directories)
- Mac and Windows users can relate a *directory* to the same idea as a *folder*

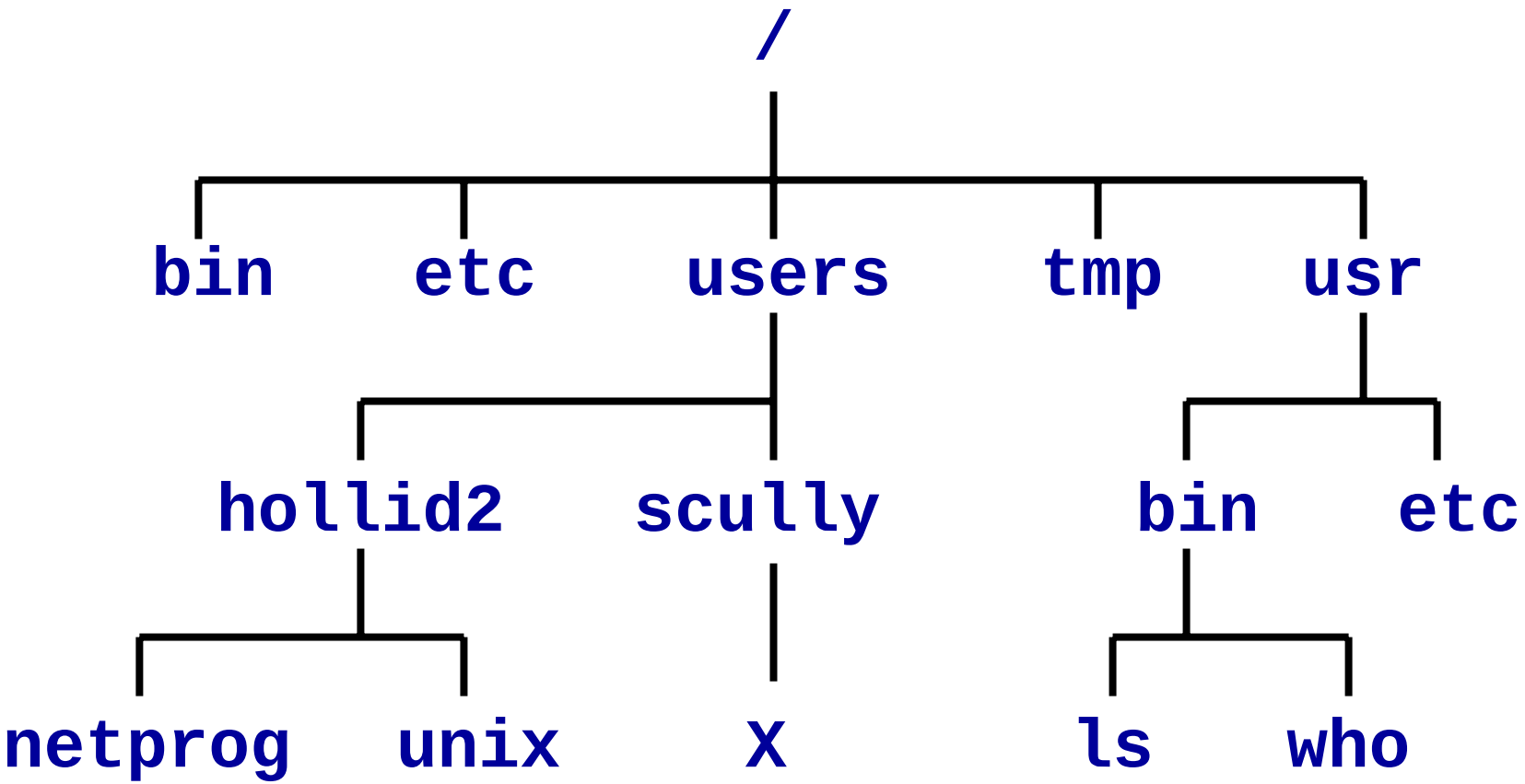
More about File Names

- Every file must have a name
- Each file in the same directory must have a unique name
- Files that are in different directories can have the same name
- Note: Unix is *case-sensitive*
  - So, “texas-fight” is different than “Texas-Fight”

Unix Filesystem

- The filesystem is a hierarchical system of organizing files and directories
- The top level in the hierarchy is called the "root" and holds all files and directories.
- The name of the root directory is / (the “slash” directory)
- Typical system directories below the root directory include:
  /bin contains many of the programs which will be executed by users
  /etc files used by system administrators
  /dev hardware peripheral devices
  /proc a pseudo file system which tracks running processes and system state (vm)
  /lib system libraries
  /usr normally contains applications software
  /home home directories for different systems

Unix Filesystem (an upside-down tree)



Pathnames

- The full *pathname* of a file includes the file name and the name of the directory that holds the file, and the name of the directory that holds the directory that holds the file, and the name of the …
  ….*all the way up up to the root directory*

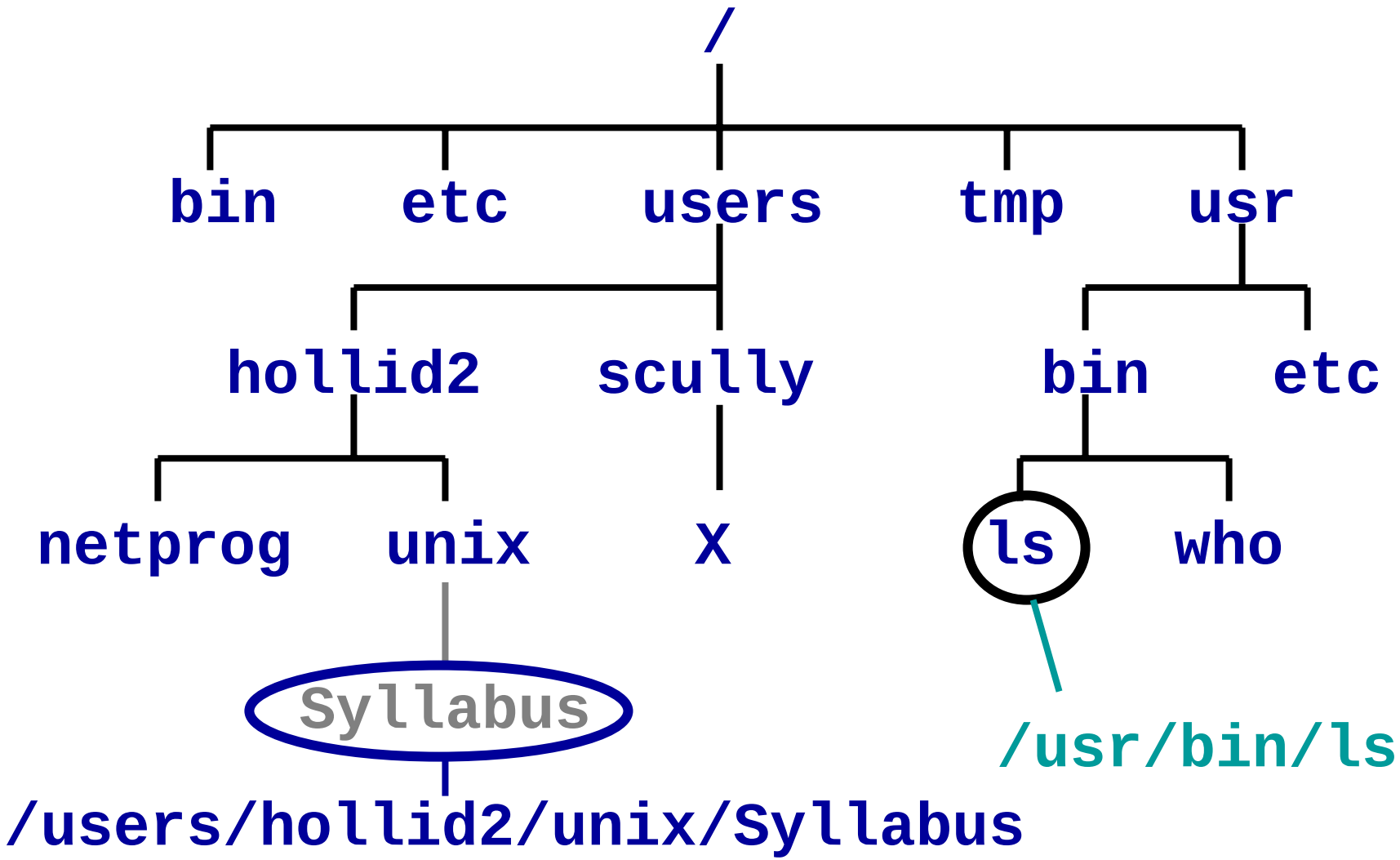
- The full pathname of every file in a Unix *filesystem* is unique (falls from the requirement that every file in the same directory must be a unique name)

Pathnames (cont.)

- To create a pathname you start at the root (so you start with "/"), then follow the path down the hierarchy (including each directory name) terminating with the filename

- In between every directory name you put a "/"

Pathname Examples

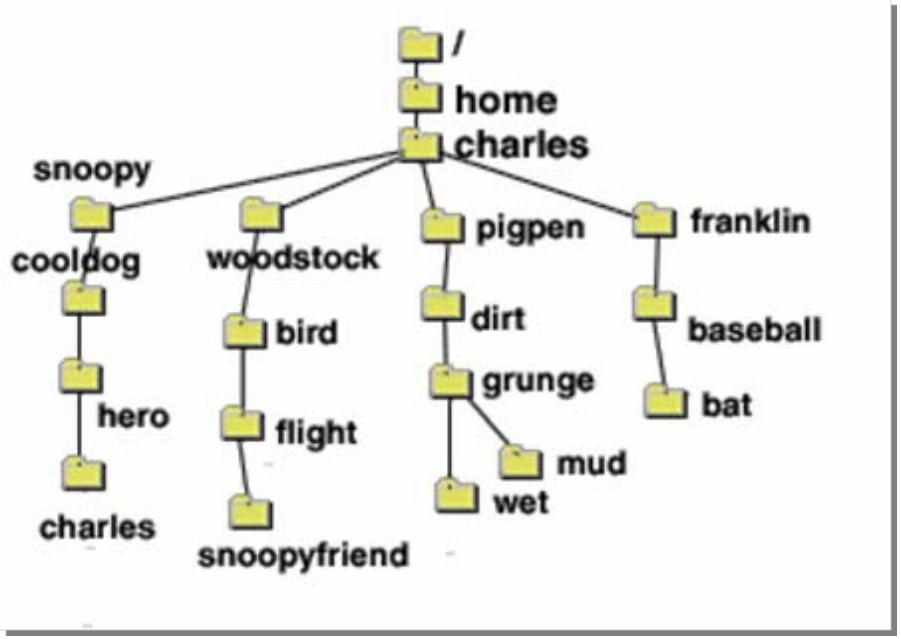


Absolute Pathnames

- The pathnames described in the previous slides start at the *root*
- These pathnames are called *absolute pathnames*
- We can also talk about the pathname of a file *relative* to a directory

Relative Pathnames

- A *relative pathname* specifies a file in relation to the current working directory (CWD)
- If CWD=*/home*, *then the relative pathname to charles is: charles*
- If CWD=*/home*, *then the relative pathname to pigpen is: charles/pigpen*
- *If CWD=/home, then the relative pathname to baseball is: charles/franklin/baseball*



- Most Unix commands deal with pathnames
- We often use relative pathnames when specifying files (for convenience)

Special Directory Names

- There is a special relative pathname for the current working directory (CWD):

  . *(yes, that's a dot)*

  Example**: ./foo** (refers to "foo" in the current directory)

- There is also a special relative pathname for the parent directory:

  .. *(affectionately known as a dot-dot)*

  Example**: ../foo** (refers to "foo" in the parent directory)

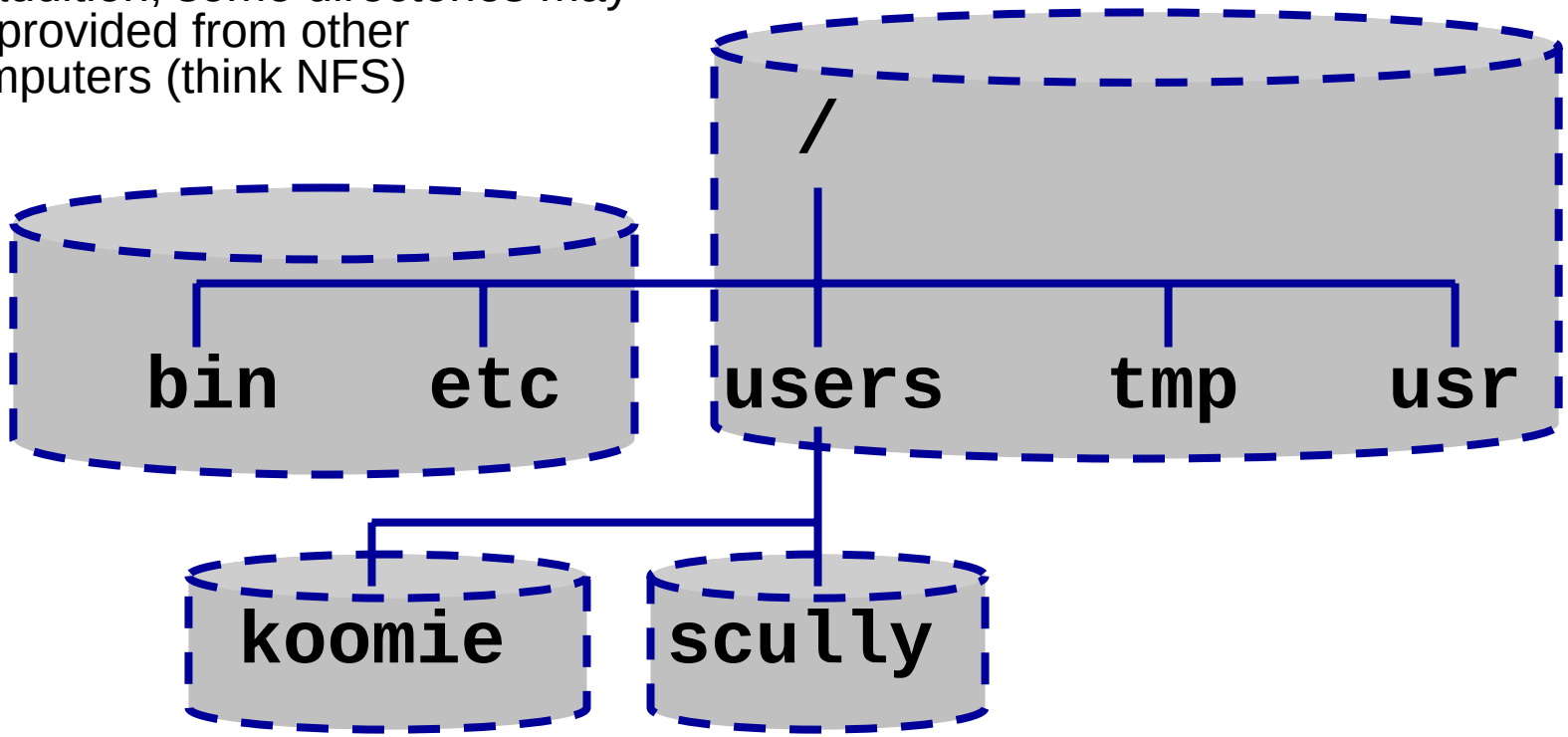
- There is a special symbol for the location of your home directory:

  ~ (that's a tilde)

  Example**: ~koomie** (refers to the home directory for user "koomie")

Disk vs. Filesystem

- Note that the file system hierarchy can actually be served by one or more physical disk drives
- In addition, some directories may be provided from other computers (think NFS)



Basic Commands

- Some basic commands for interacting with the Unix file system are:

– ls	- pwd	- touch
– cd	- cp	- mkdir
– df	- awk	- rmdir
– cat	- rm	- find
– more (less)	- chmod	- grep
– head	- tail	- chown/chgrp

- We will focus on ls first

The `ls` command

- The ls command displays the names of files
- If you give it the name of a directory as a *command line parameter* it will list all the files in the named directory

Example `ls` Commands

`ls`	list files in current directory
`ls /`	list files in the root directory
`ls .`	list files in the current directory
`ls ..`	list files in the parent directory
`ls /usr`	list files in the directory **`/usr`**

Command Line Options

- We can modify the output format of the ls program with a *command line option*.
- The ls command supports a bunch of options:
  - l *long* format (include file times, owner and permissions)
  - a *all* (shows hidden files as well as regular files)
  - F include special char to indicate file types

In Unix, hidden files have names that start with "."

`ls` Command Line Options

- To use a command line option precede the option letter with a minus:

  `ls -a` or `ls -l`

- You can use two or more options at the same time like this:

  `ls -al`

General `ls` command line

- The general form for the ls command is:

  `ls [options] [names]`

- The options must come first!

- You can mix any options with any names.

- An example:

  `ls -al /usr/bin`

Command Line Syntax

- `ls [options] [names]`
  - The brackets around options and names in the general form of the ls command means that something is optional
  - This type of description is common in the documentation for Unix commands
  - Some commands have required parameters

Variable Argument Lists

- You can give the ls command many files or directory names to display:

```
ls /usr /etc
ls -l /usr/bin /tmp /etc
```

Where to Get More Information?

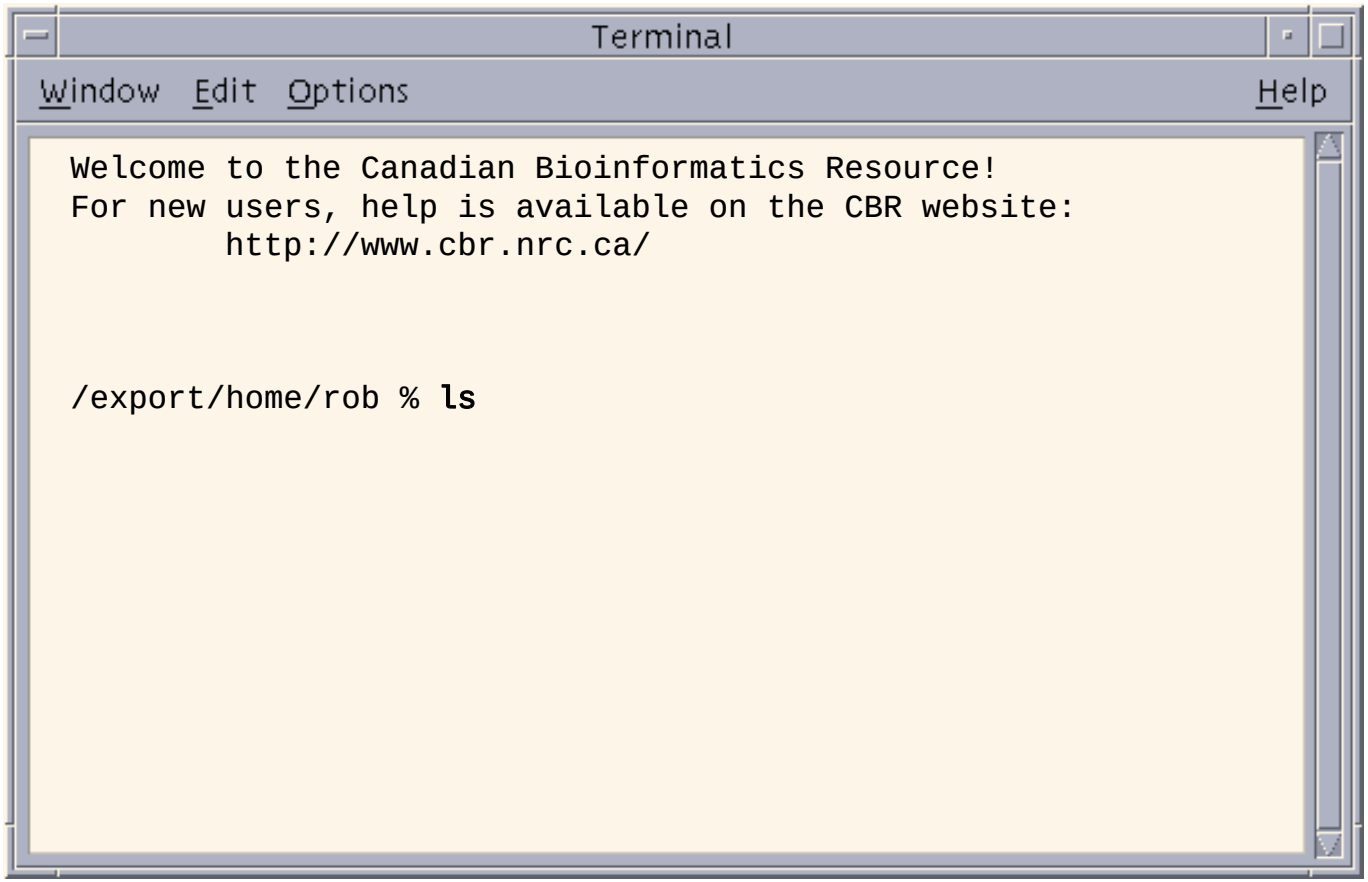
- Almost all UNIX systems have extensive *on-line* documentation known as **man pages** (short for "manual pages").
- The Unix command used to display them is man. Each page is a self-contained document.
- So, to learn more about the ls command, refer to it's man page:
  - **man ls**
- Man pages are generally split into 8 numbered sections (on BSD Unix and Linux):
  - 1 General commands
  - 2 System calls
  - 3 C library functions
  - 4 Special files (usually devices, those found in /dev)
  - 5 File formats and conventions
  - 6 Games
  - 7 Miscellaneous
  - 8 System administration commands and daemons
- You can request pages from specific sections:
  - **man 3 printf** (shows manpage for C library function)

Example Man Page

```
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-c|-w|-tZ]  [-H[browser]] [-T[device]] [-adhu7V] [-i|-I] [-m sys-
       tem[,...]] [-L locale] [-p string] [-C file] [-M path] [-P  pager]  [-r
       prompt] [-S list] [-e extension] [[section] page ...] ...
       man -l [-7] [-tZ] [-H[browser]] [-T[device]] [-p string] [-P pager] [-r
       prompt] file ...
       man -k [apropos options] regexp ...
       man -f [whatis options] page ...

DESCRIPTION
       man is the system's manual pager. Each page argument given  to  man  is
       normally  the  name of a program, utility or function.  The manual page
       associated with each of these arguments is then found and displayed.  A
       section,  if  provided, will direct man to look only in that section of
       the manual.  The default action is to search in all  of  the  available
       sections, following a pre-defined order and to show only the first page
       found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the
 Manual page man(1) line 1
```

UNIX Command Examples

Type UNIX commands at the prompt:



UNIX Command Examples

Type UNIX commands at the prompt:

Terminal

Window Edit Options Help

```
Welcome to the Canadian Bioinformatics Resource!
For new users, help is available on the CBR website:
        http://www.cbr.nrc.ca/



/export/home/rob % ls
bin/          my_data/         data.fasta
/export/home/rob %
```

UNIX Command Examples

Type UNIX commands at the prompt:

Terminal

Window Edit Options Help

```
/export/home/rob % ls -l
total 26
drwxr-xr-x   2 rob      users        512 Nov  2 15:09 bin/
drwx------   2 rob      users        512 Nov  1 09:19 my_data/
-rw-------   1 rob      users        343 Dec  5 13:51 data.fasta
```

UNIX Command Examples

Parameters can be *flags* or *arguments*:

Terminal

Window Edit Options Help

```
/export/home/rob % ls –l data.fasta
-rw-------   1 rob      users        343 Dec  5 13:51 data.fasta
```

UNIX Command Examples

Parameters can be *flags* or *arguments*:

Terminal

Window Edit Options Help

```
/export/home/rob % ls -l data.fasta
-rw-------   1 rob      users        343 Dec  5 13:51 data.fasta
```

UNIX Command Examples

Parameters can be *flags* or *arguments*:

Terminal

Window Edit Options Help

```
/export/home/rob % ls -l data.fasta
-rw-------   1 rob      users        343 Dec  5 13:51 data.fasta
```

UNIX Command Examples

Parameters can be *flags* or *arguments*:

Terminal

Window Edit Options Help

```
/export/home/rob % ls -l data.fasta
-rw-------   1 rob      users        343 Dec  5 13:51 data.fasta
```

Flags change the behaviour of the command.
Arguments define the scope of the command.

Unix: A Culture in Itself

"Two of the most famous products of Berkeley are LSD and Unix. I don't think that this is a
(Anonymous quote from The UNIX-HATERS Handbook.)

References/Acknowledgements

- National Research Council Canada (Rob Hutten, Canadian Bioinformatics Resource)
- *Intro. to Unix*, Dave Hollinger, Rensselaer Polytechnic Institute
- Top 500 Supercomputing Sites: (www.top500.org)
- The Open Group (www.unix.org)
- History of Linux (www.linux.org/info/linux_timeline.html)
- Cygwin (http://www.cygwin.com/)
- XFree86 (http://www.xfree86.org/)
- CentOS (www.centos.org)