3-Unix
History, Files, Paths

CSC-60

Unix History

A brief history of UNIX OS

- The Unix OS was developed (based on Multics & CTSS operating systems) by Ken Thompson at the AT&T Bell Laboratories in 1969. He wanted to create an multi-user operating system to run “space wars” game.

- Ken’s philosophy was to create an operating system with commands or “utilities” that would do one thing well (i.e. **UNIX**). Pipes could be used combine commands...

History of Unix OS

- The first versions of UNIX were written in “machine-dependent” program (such as PDP-7).

- Ken Thompson approached Dennis Ritchie, developer of C language, and in 1973 they compiled UNIX in C to make operating system “portable” to other computers systems.

History of Unix OS

PDP-7 Machine

Ken (seated) and Dennis (standing) at a PDP-11 in 1972.

Development of Unix OS

Students at University of California (in Berkley) further developed the UNIX operating system and introduced the BSD version of Unix

Unix

Bell Labs
UNIX System V (5)
Proprietary (Commercial Implementations)

Berkley Software Distribution (BSD - 1979) Free

- 1981: Berkeley UNIX 4.1 BSD (enhanced with vi, csh, and virtual memory management)
- 1983: Berkeley UNIX 4.2 BSD (added TCP/IP networking, sockets and a new file system)

Development of Unix OS

There were versions of UNIX for the Personal Computer (PC), such as XENIX, etc., but they didn't catch on in popularity until Linux was developed in the early 90's.

History of Linux

- Linux operating system developed by programming student Linus Torvalds (1991)

- Linus wanted to develop Unix-like OS <u>just</u> to experiment with new 386 computer at the time...

- Linus invited other programmers to improve the Kernel. Overtime, it was ported to various hardware architectures

GNU (GNU 's NOT UNIX) Project

Richard Stallman

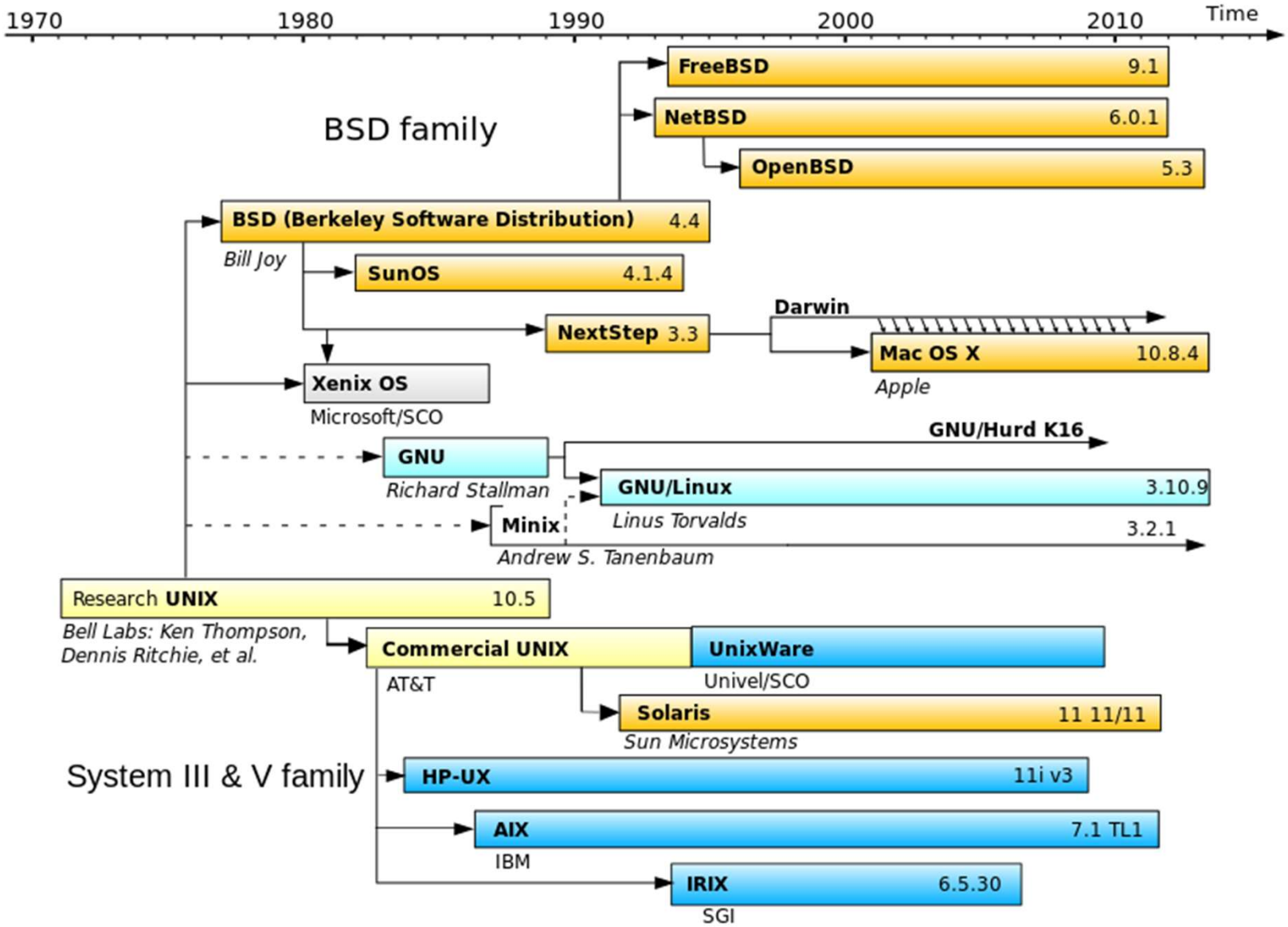
- Launched in 1984 to develop a complete UNIX Operating system that is free software: www.gnu.org (Free Software Foundation - founded by **Richard Stallman**)

  Usually C code, but some C++

- GNU has a 72-page document of coding standards – well-written code. It provides useful tools such as Emacs, Gcc, Bash shell, glibc (C library)

- GNU's kernel (Called Hurd) was not working (not stable).

Brief history of Unix-like operating systems



Source: http://en.wikipedia.org/wiki/Linux

Standardizations (1 of 2)

- POSIX: Portable Operating System Interface
  - An IEEE-standard which describe the behavior of UNIX and UNIX-like OS.
  - POSIX support assures code portability between systems and is increasingly mandated for commercial applications and government contracts.

Standardizations (2 of 2)

- SUSv3: Single UNIX Specification version 3
  - Beginning in 1998,joint working group known as the Austin Group began to develop the combined standard that would be know as the Single UNIX Specification Version 3 and as POSIX:2001. This name serves as referenced points throughout the book.

- Examples from our main textbook:
  - …This 65-character set, {-._a-zA-Z0-9], is referred to in **SUSv3** as the *portable filename character set*. Page 28.
  - …And Standard system defined by SUSv3… Page 43.

Operating System

An operating system is a control program for a computer that performs the following operations:

- allocates computer resources
- schedules tasks
- provides a platform to run application software for users to accomplish tasks
- provides an interface between the user & the computer

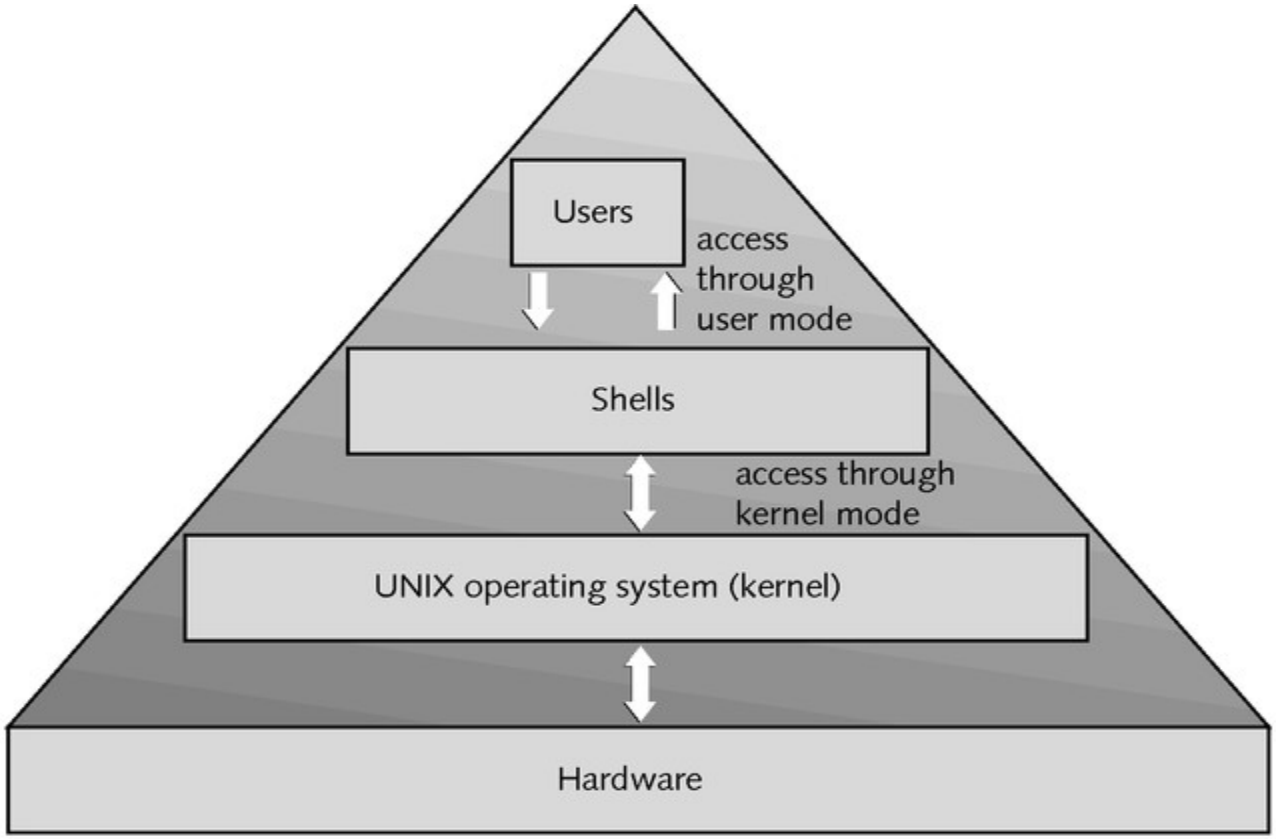


Figure 1-5 Layers of a UNIX system

Shell as a user interface

- A shell is a command interpreter, an interface between a human (or another program) and the OS
  - runs a program, perhaps the **ls** program.
  - allows you to edit a *command line*.
  - can establish alternative sources of input and destinations for output for programs.
- **ls**, itself, just another program

Kernel (OS)

- Interacts directly with the hardware through device drivers
- Provides sets of services to programs, insulating these programs from the underlying hardware
- Manages memory, controls access, maintains file system, handles interrupts, allocates resources of the computer
- Programs interact with the kernel through *system calls*

UNIX Software Architecture

Application User's Interface (AUI)

Application Programmer's Interface (API)

Operating System

Applications: Compilers, word processors, spreadsheets, ftp, telnet, Web browser, etc.

UNIX shell:

Language libraries: C, C++, Java, FORTRAN, etc.

System call interface (entry points to kernel)

UNIX kernel:

- File management
- Primary and secondary storage management
- Interprocess Communication (IPC)
- Process Management
- CPU scheduler

Device Drivers: Mouse driver, printer driver, CD-ROM driver, hard disk driver, etc.

Hardware: Wires, capacitors, resistors, transistors, ICs, mouse, display monitor, keyboard, CPU, RAM, hard disk, CD_ROM, printer, etc.

Source:
UNIX the textbook.
By Sarwar, Koretsky, Sarwar.

Linux Files

File Attributes

File attributes

- Every file has some attributes:
  - Access Times:
    - when the file was created
    - when the file was last changed
    - when the file was last read
  - Size
  - Owners (user and group)
  - Permissions
  - Type – directory, link, regular file, etc.

File Owners

- Each file is owned by a user.
- You can find out the username of the file's owner with the -l or -o option to ls:

athena.ecs.csus.edu - PuTTY

```
[bielr@athena csc60]> ls -l
total 540
-rw-------  1 bielr faccsc      2 Dec 21 12:58 >
-rwx------  1 bielr faccsc   6438 Sep 15 13:47 a.out*
drwx------  2 bielr faccsc   4096 Oct 21 09:34 ClassExamples/
-rw-------  1 bielr faccsc    138 Dec 22 09:39 lsout
drwx------  5 bielr faccsc   4096 Jan 24 13:08 mywork/
drwx------  6 bielr faccsc   4096 Dec 18 15:58 myworkf16/
drwx------  8 bielr faccsc   4096 Dec 16 15:07 myworkS16/
-rwx------  1 bielr faccsc   6438 Sep 19 09:04 reverse*
-rw-------  1 bielr faccsc    993 Sep 16 13:24 reverse1.c
drwx------  2 bielr faccsc   4096 Jan 15 13:01 student/
-rw-------  1 bielr faccsc    527 Nov 16 08:35 testScript.txt
-rw-------  1 bielr faccsc 235289 Apr 17  2016 tlpi-160401-dist.tar.gz
drwx------ 48 bielr faccsc   4096 Nov 10 09:26 tlpi-dist/
-rw-------  1 bielr faccsc 252898 Sep 21 15:26 trylab1.txt
-rw-------  1 bielr faccsc     12 Dec 22 09:40 wcout
[bielr@athena csc60]>
```

File Attributes shown by **ls –l**

drwx------ 2 bielr faccsc 4096 Apr 27 15:43 ClassExamples/

Field	Meaning
First letter of first field	File type: **-** ordinary file **b** block special file **c** character special file **d** directory **l** link **p** named pipe (FIFO) **s** socket

File Attributes shown by **ls –l**

drwx------ 2 bielr faccsc 4096 Apr 27 15:43 ClassExamples/

Field	Meaning
Remaining letters of first field	Access permissions for owner, group, and others (r w x)
Second field	Number of links

File Attributes shown by **ls –l**

```
drwx------  2 bielr faccsc   4096 Apr 27 15:43 ClassExamples/
```

Field	Meaning
Third field	Owner's login name
Fourth field	Owner's group name (can also be a number)

File Attributes shown by **ls –l**

drwx------ 2 bielr faccsc 4096 **Apr 27 15:43** ClassExamples/

Field	Meaning
Fifth field	File size in bytes
Sixth, seventh, and eighth field	Date and time of last modification
Ninth field	File name

ls -l

```
$ ls -l foo
-rw-rw----   1 bielr  faccsc 13 Jan 10 23:05 foo
```

permissions

owner

group

size

time

name

File Permissions

- Each file has a set of permissions that control who can work with the file.
- There are three *types* of permissions:
  - read abbreviated **r**
  - write abbreviated **w**
  - execute abbreviated **x**
- There are 3 *sets* of permission:
  - user
  - group
  - other (the world, everybody else)

`ls -l` and permissions

`-rwxrwxrwx`

User Group Others

Type of file:
- – plain file
d – directory
s – symbolic link

rwx

- Files:
  - **r -** allowed to read.
  - **w -** allowed to write
  - **x -** allowed to execute
- Directories:
  - **r -** allowed to see the names of the file.
  - **w -** allowed to add and remove files.
  - **x -** allowed to enter the directory

Changing Permissions

- The **chmod** command changes the permissions associated with a file or directory.
- There are a number of forms of chmod, this is the simplest:
  - chmod mode file

chmod – numeric modes

- Consider permission for each set of users (user, group, other) as a 3-bit #
  - r – 4
  - w – 2
  - x – 1
- A permission (mode) for all 3 classes is a 3-digit octal #
  - 755 – rwxr-xr-x (user: read/write/execute, group:read/execute, others:read/execute)
  - Example: > chmod 755 lab1.c
  - 644 – rw-r—r– (user: read/write, group:read, others:read)
  - 700 – rwx------ (user: read/write/execute, group: no access, others: no access)

chmod – symbolic modes

- Can be used to set, add, or remove permissions
- Mode has the following form:
  - **[ugoa][+-=][rwx]**
- u – user g – group o – other a – all
  - + add permission
  - - remove permission
  - = set permission

chmod examples

$ **ls -al foo**

-rwx rwx --x 1 hollingd grads foo

$ **chmod g-wx foo**

$ **ls -al foo**

-rwxr----x 1 hollingd grads foo

$ **chmod u-r .**

$ **ls**

ls: .: Permission denied

CSC-60

Unix File System

Home Directory

- The user's personal directory. E.g.,
  - /gaia/class/student/xyz
  - /gaia/class/student/yzx
- Where all your files go (hopefully organized into subdirectories)
- Mounted from a file server – available (seamlessly) on *any* department machine you log into

  (athena, sp1, sp2, sp3, atoz)

Home Directory

- Your *current directory* when you log in
- cd (by itself) takes you home
- Location of many startup and customization files. E.g.:
  - .vimrc .bashrc .bash_profile .forward .plan .mozilla/ .elm/ .logout

Directories

- A directory is a special kind of file - Unix uses a directory to hold information about other files and directories.

- We often think of a directory as a container that holds other files (or directories).

- A directory is the same idea as a *folder* on Windows.

More about File Names

- Review: every file has a name (at least one).
- Each file *in the same directory* must have a unique name.
- Files that are in different directories can have the same name.

File Time Attributes

- Time Attributes:
  - when the file was last changed **ls -l**
  - sort by modification time **ls -lt**

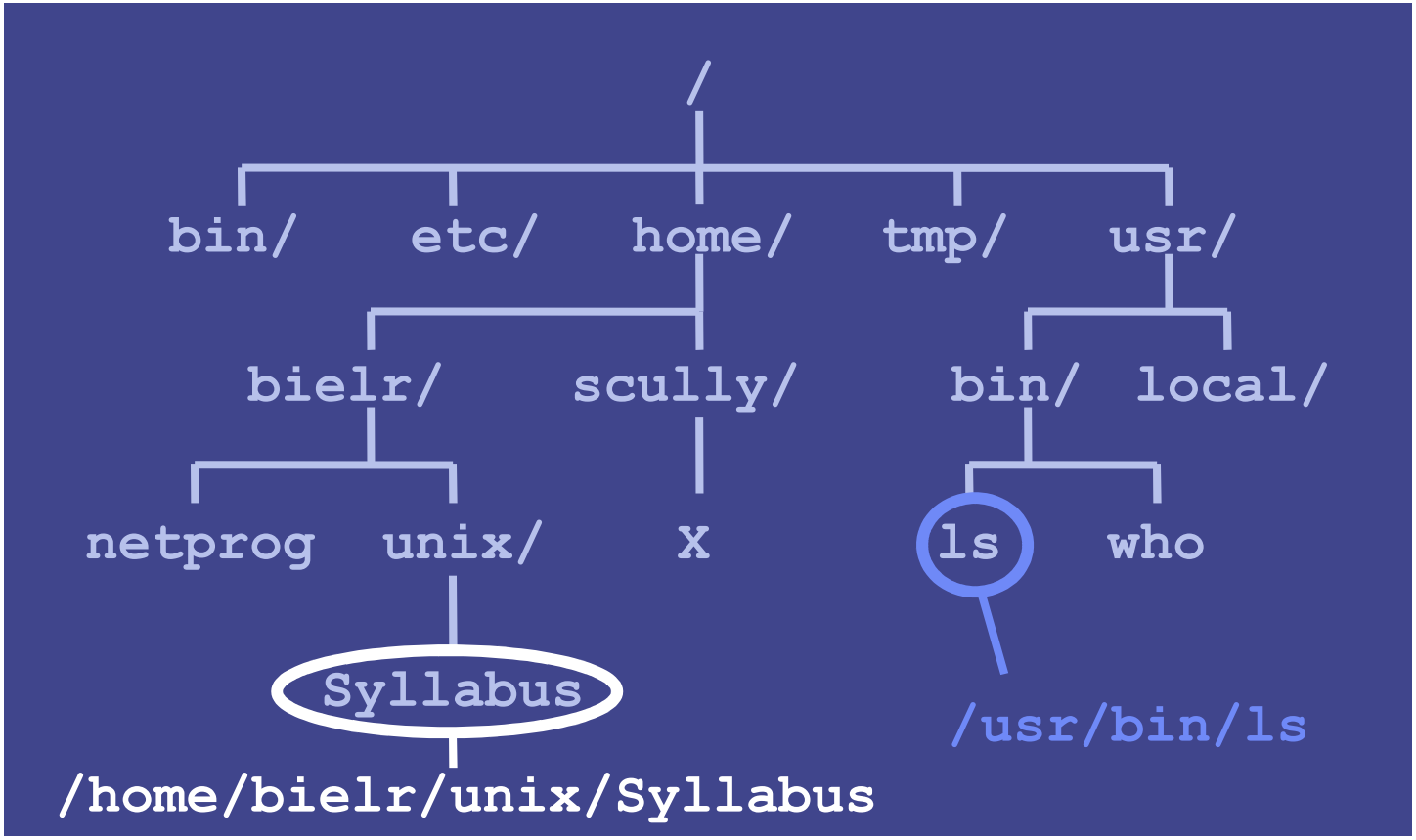
Unix Filesystem

- 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 in the filesystem.

- The **name** of the root directory is **/**

Pathname Examples



/home/bielr/unix/Syllabus is the pathname

Pathnames

- The *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 … up to the root.

- The pathname of every file in a given *filesystem* is unique.

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) and you end with the filename.

- In between every directory name, we use a delimiter of "/".

Absolute Pathnames

- The pathnames described in the previous slides start at the *root*.
- These pathnames are called "absolute pathnames".

Relative Pathnames

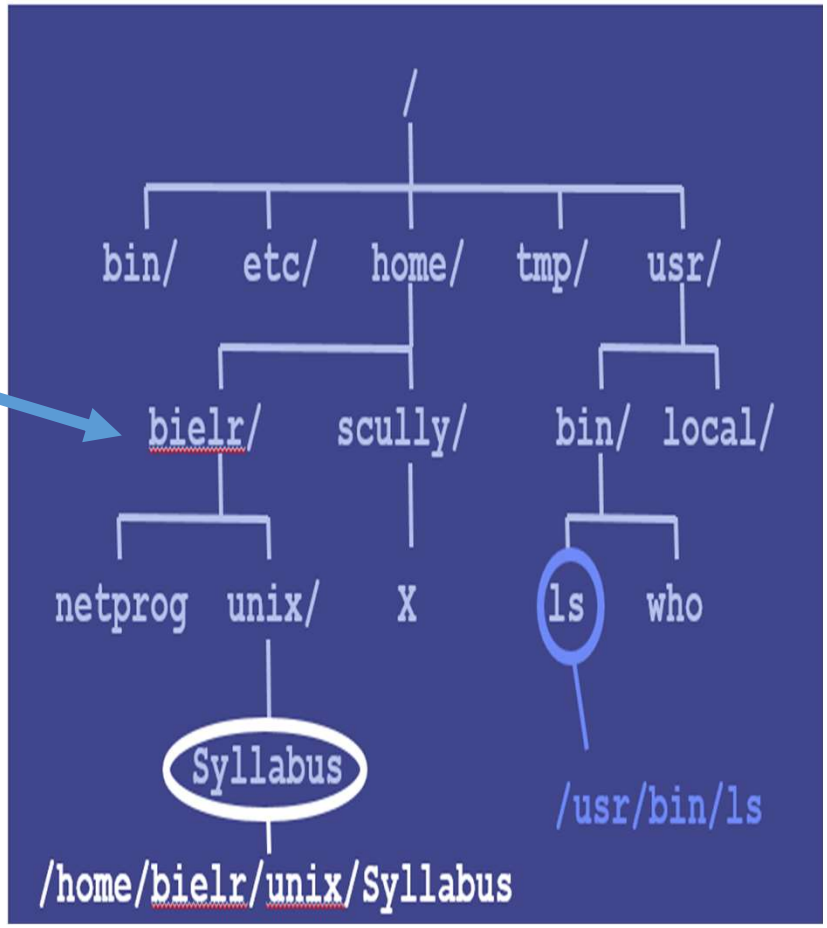
Prefixed w/the current directory, $PWD
So, **relative** to the current working directory

$ **cd /home/bielr**
$ **pwd**
/home/bielr *(current working dir)*

$ **ls unix/Syllabus** *(relative pathname)*
unix/Syllabus

$ **ls X**
ls: X: No such file or directory (?)

$ **ls /home/scully/X**
/home/scully/X *(found it!)*



Special Relative paths…

. The current directory

.. The *parent* directory

$ **pwd**

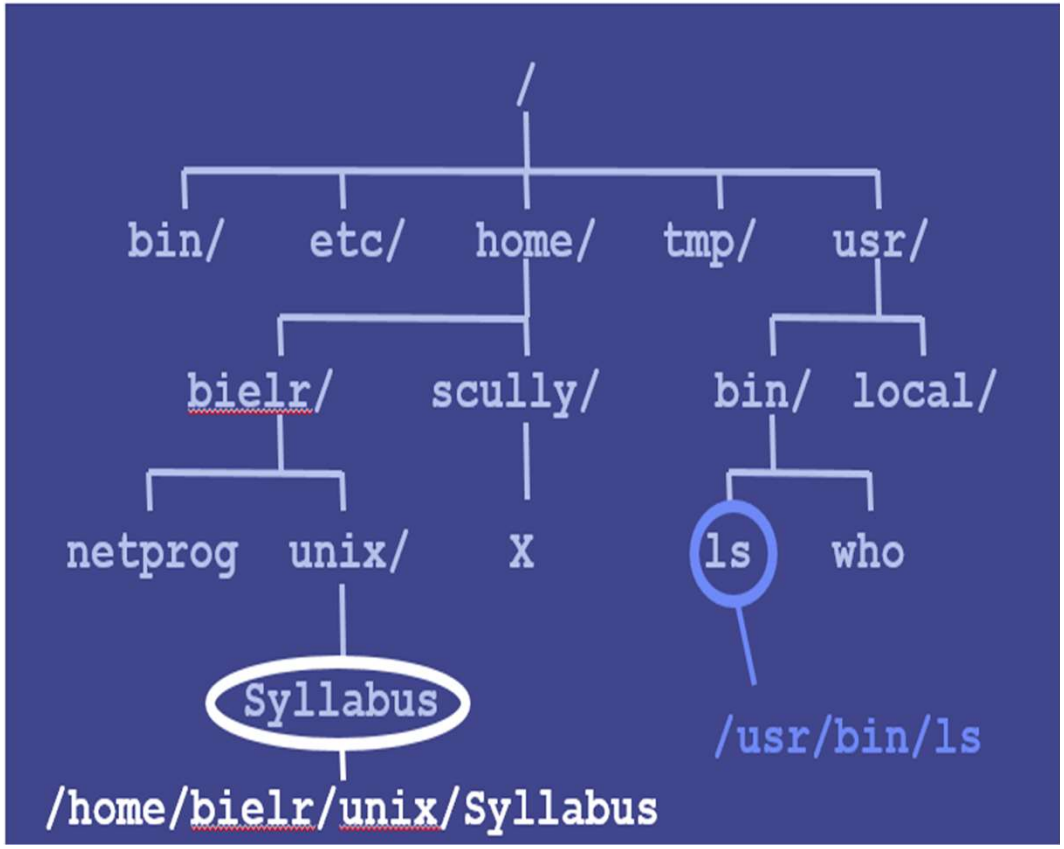
/home/bielr

$ **ls ./netprog**

./netprog

$ **ls ../scully**

X



Some Standard Directories & Files 1 of 2

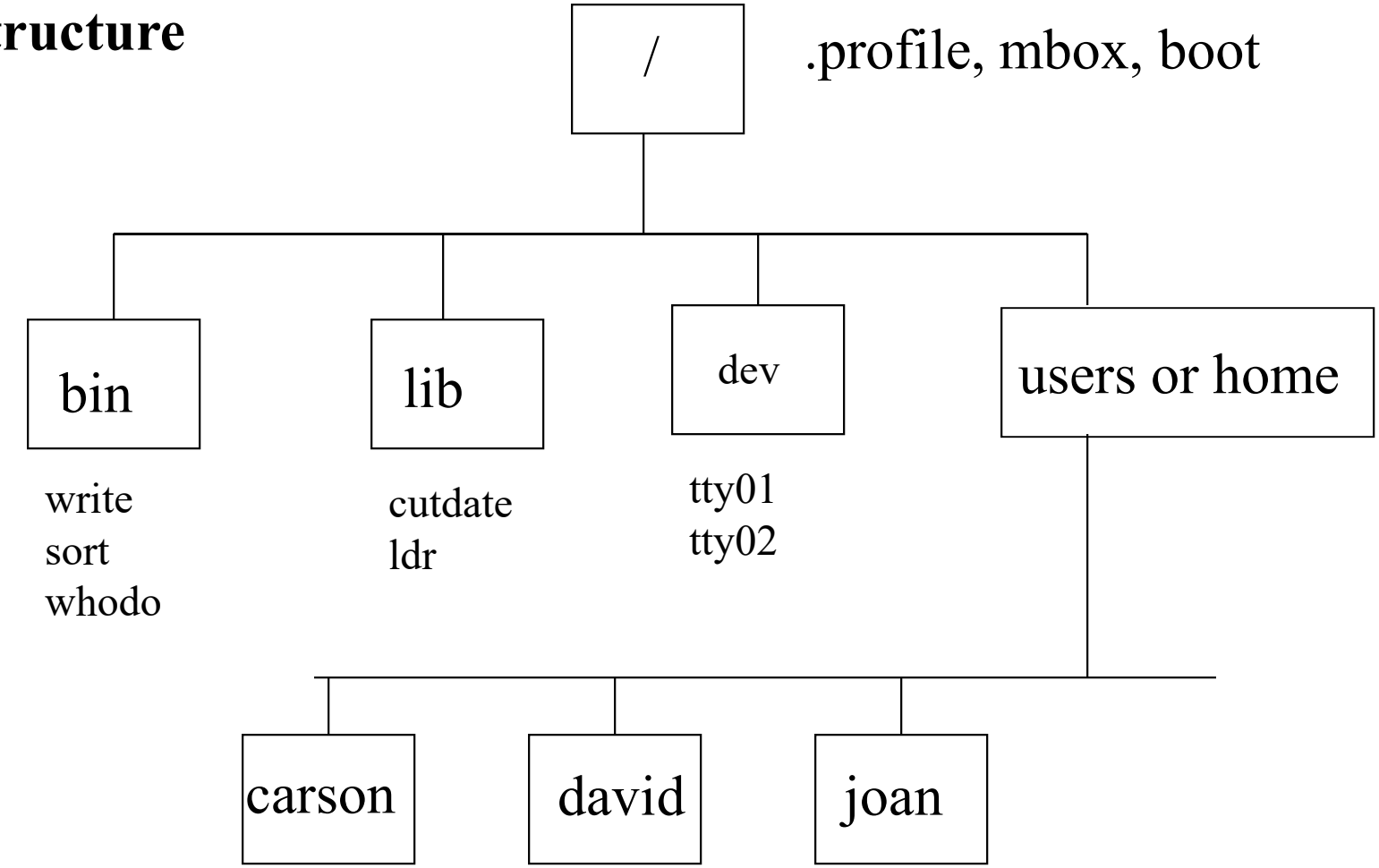
- **Root Directory** (**/**). The top of the file system
- **/bin**. The binary directory. Contains binary (executable) images of most UNIX/Linux commands.
- **/dev**. The device directory. Has files corresponding to all the devices connected to the computer.
- **/etc**. Contains commands and file for system administration. The typical user is usually not allowed to use these commands and files.
- **/lib**. The library directory. Contains a collection of related files for a given language in a single file called **archive**. Many UNIX/Linux systems contain libraries for C, C++, and FORTRAN.

Some Standard Directories & Files 2 of 2

- **/tmp**. Contains temporary files. The system admin determines the length of their life, usually only a few minutes.
- **/users**. The home directory for all the users on a system.
- **/usr**. UNIX System Resource. Contains subdirectories: utilities, tools, language libraries, manual pages.

Directories & Files

LINUX directory structure



Each of these user then have their own files.

The **root** directory is always named “/”.

Parent directory is the one directly above another directory.

Child directory is a directory directly below another directory.

Sub-directory – another name for a **child** directory.

When you log on, you are put into your **home** directory.

>pwd /* **p**ath of the **w**orking **d**irectory */

*Result of a **pwd** in my own directory:*

```
[bielr@athena ~/csc60]24> pwd
/gaia/home/faculty/bielr/csc60
```

Notice the full path name, starting with "/" for the root.

```
[doej@athena/21> pwd
/gaia/class/student/doej
```

To return to your home directory:

> **cd** /* **C**hange **D**irectory */

To go to a sub-directory:

> **cd** *directory-name*

From my home directory, to get to the sub-directory:

> **cd** csc60

An alternate to **cd** is **chdir.**

Forming File Names:

All UNIX/Linux systems can handle file names of up to 14 characters. Some can use names as long as 256.

Names are formed from:

A to Z
a to z
0 to 9
_ (underscore)
. (period)
, (comma)

NO slash. Better not to use Space or Dash.

Creating a Directory:

mkdir

Example:

> **mkdir** csc60

Renaming Directories:

You must first be in the parent directory.

>mv *original-name new-name*

Removing Directories:

>rmdir *directory-name*

Ambiguous File Names:

? Represents any other character
* Represents no character or any number of characters.

<u>Examples</u>:

***a** All files with names ending in **a**

***[xyz]** All files with names ending in **x**, **y**, or **z.**

***.?** All files that contain a period with exactly one character following.

?? All files with two-character names.

***.obj** All files with **.obj** as the last four characters.

3-Unix
History, Files, Paths

The End