

Basic UNIX

- users /etc/passwd (/etc/master.passwd)
- passwords
- groups /etc/group
- shell
 - sh - Bourne Shell (AT&T) (Steve Bourne)
 - csh - C shell (Berkeley)
 - ksh - Korn shell (Sys V) (David Korn)
 - bash - Bourne Again Shell (GNU)
 - Konqueror/Dolphin/File Manager,
- directory - (Windows calls them "folders")
 - Home directory: /home/phil
 - Working directory: /home/phil/Class/cs347/lec
 - Root directory: /
- path -- name of file or directory
 - relative: dir1/file.name
 - absolute: /rootdir/dir2/dir3

- Standard entries in all directories

- . -- the directory itself
 - .. -- the directory's parent

- Man pages -- UNIX documentation

- Sections -- "chapters"

- Section 1 -- user commands
 - Section 2 -- system calls
 - Section 3 -- library calls
 - Section 4 -- special files
 - Section 5 -- file formats
 - Section 6 -- games
 - Section 7 -- macros & packages
 - Section 8 -- system administration
 - Section 9 -- system documentation (NetBSD and others)

- Convention: name(section-number)

- ls(1)
 - fork(2)

☐ More man pages ...

- ☐ more(1)

- ☐ mkdir(1)

- ☐ rm(1)

- ☐ ...

☐ UNIX Philosophy

- ☐ "A tool for each Job"

☐ C is standard language of UNIX

System Calls

- request to OS
- typically a trap instruction
- often an assembly wrapper
- most languages have access to call system calls
- Our use ... like C function calls
- `int read (int fd, void *buf, size_t nbytes)`
- Return values
 - return information about system call
 - many have a value representing an error condition
- System call errors
 - access to which error via "errno" variable
 - `#include <errno.h>`
 - `/usr/include/errno.h` -- list of errors (`/usr/include/sys/errno.h`)
 - `strerror(3)` -- `errno` -> string
 - `perror(3)` -- prints `errno` string
 - Don't use for "User errors", only system call errors
 - May use for some library routines that call system calls

Files

"Everything in UNIX is a file!"

- /dev/console, /dev/wd0a (NetBSD)

- network connections

- File vs File Descriptor (aka FD)

 - Open(2) returns fd

 - Other system calls use fd

 - fd -- small non-negative integer

 - index in a kernel table of "open files"

 - Standard "files" fd - (stdio-name, buffering)

 - 0 - (stdin, buffered)

 - 1 - (stdout, buffered)

 - 2 - (stderr, unbuffered)

 - System calls -- open, read, write, lseek, close

- Redirection in shells

 - ls > file_list # redirect stdout

 - ush < script >> output 2> errs # redirect all three

 - > create an empty file, >> append to existing file

Programs vs. Process

☐ Program

- ☐ source code (file)
- ☐ executable (file)

☐ Process

- ☐ a unique execution of code
 - ☐ May execute several programs in 1 process
- ☐ has a unique identifier -- pid
- ☐ System Calls -- fork, execve, wait, getpid, getuid, ...

☐ Other ID's in UNIX

- ☐ pid -- process id (getpid(2))
- ☐ uid -- user id (getuid(2))
- ☐ gid -- group id (getgid(2))

Other things

☐ Signals

- ☐ method to notify a process of an event

- ☐ actions --

 - ☐ catch

 - ☐ ignore

 - ☐ default

- ☐ common signals

 - ☐ SIGINT -- C-c (Control-C)

 - ☐ SIGTSTP -- C-z

 - ☐ SIGSEGV -- program error

☐ Time

- ☐ seconds since 00:00:00 1 Jan 1970

- ☐ 32 bit number runs out in 2038 (31 bits)

- ☐ NetBSD and Linux have converted kernel to 64 bit number

 - ☐ 292271023045 years after 1970

☐ Standards

- ☐ ANSI C -- Language
- ☐ POSIX -- IEEE - for UNIX
- ☐ XPG3 -- vendors - for UNIX

☐ System Types

- ☐ #include <sys/types.h>
- ☐ size_t
- ☐ uid_t
- ☐ pid_t
- ☐ gid_t
- ☐ clock_t
- ☐ ...

