Advanced Unix/Linux System Program
Instructor: William W.Y. Hsu
CONTENTS

› Course preliminaries
› Introduction
› Unix history
› Unix basics
About this class

› The class is called “Advanced Unix/Linux System Programming”.

› It is not:
  – an introduction to Unix
  – an introduction to programming
  – an introduction to C (or C++)
In a nutshell: the “what”
In a nutshell: the “what”
In a nutshell: the “what”

› Gain an understanding of the UNIX operating systems.
› Gain (systems) programming experience.
› Understand fundamental OS concepts (with focus on UNIX family): multi-user concepts.
   – Basic and advanced I/O
   – Process relationships
   – Interprocess communication
   – Basic network programming using a client/server model
In a nutshell: the “why”

› Understanding how UNIX works gives you insights in other OS concepts.

› System level programming experience is invaluable as it forms the basis for most other programming and even use of the system.

› System level programming in C helps you understand general programming concepts.

› Most higher level programming languages (eventually) call (or implement themselves) standard C library functions.
In a nutshell: the “how”

```c
static char dot[] = ".", *dotav[] = {dot, NULL};
struct winsize win;
int ch, fts_options;
int kflag = 0;
const char *p;

setprogname(argv[0]);
setlocale(LC_ALL, "");

/* Terminal defaults to -Cq, non-terminal defaults to -l. */
if (isatty(STDOUT_FILENO))
{
    if (ioctl(STDOUT_FILENO, TIOCGWINSZ, &win) == 0 && win.ws_col > 0)
        termwidth = win.ws_col;
    f_column = f_nonprint = 1;
}
else
    f_singlecol = 1;

/* Root is -A automatically. */
if (!getuid())
    f_listdot = 1;

fts_options = FTS_PHYSICAL;
while ((ch = getopt(argc, argv, "1ABCFLRSTWabcdfghiklmnopqrstuvwxyz")) != -1)
{
    switch (ch)
    {
    /* The -l, -C, -1, -m and -x options all override each other so shell
        aliasing works correctly. */
    case '1':
        f_singlecol = 1;
    }
```
In a nutshell: the “how”

$ $EDITOR cmd.c
$ cc -Wall -g -o cmd cmd.c
$ ./cmd

$ echo "Hooray!" Hooray!
$
In a nutshell: the “how”

› Open your laptops!
  – Ignore Facebook, Twitter, Email, Line, ...

› Now compile and run this program:
  http://course.deepsea9.taipei/linux/welcome.c
In a nutshell: the “how”

$ $EDITOR cmd.c
$ cc -Wall -g -o cmd cmd.c
cmd.c: In function `main':
cmd.c:19: error: parse error before "return"
$
In a nutshell: the “how”

$ $EDITOR cmd.c

$ cc -Wall -g -o cmd cmd.c cmd.c: In function `main':

  cmd.c:19: error: parse error before "return"

$ $EDITOR cmd.c

$ cc -Wall -g -o cmd cmd.c

$ ./cmd

  Memory fault (core dumped)

$
In a nutshell: the “how”

$ $EDITOR cmd.c
$ cc -Wall -g -o cmd cmd.c cmd.c: In function `main':
  cmd.c:19: error: parse error before "return"
$ $EDITOR cmd.c
$ cc -Wall -g -o cmd cmd.c
$ ./cmd
Memory fault (core dumped)
$ echo "!@#!@!!!??#@!"
!@#!@!!!??#@!
$ gdb ./cmd cmd.core
Program terminated with signal 11, Segmentation fault.
Loaded symbols for /usr/libexec/ld.elf_so
#0 0xbbbc676a in  findenv () from /usr/lib/libc.so.12
(gdb)
If programming languages were weapons

C is an M1 Garand standard issue rifle, old but reliable.
Text and reference book

  – http://www.apuebook.com/

About Richard Stevens

› On Wikipedia:

  “Stevens died in 1999, at the age of 48. In 2000, he was posthumously awarded the USENIX Lifetime Achievement Award.”
Text and reference book

- Linux System Programming, Robert Love, O'Reilly Media, 2007
Syllabus (Tentative)

› Introduction, UNIX history, UNIX Programming Basics File I/O, File Sharing
› Files and Directories
› Filesystems, System Data Files, Time & Date
› UNIX tools: make(1), gdb(1), revision control, etc. Process Environment, Process Control
› Process Groups, Sessions, Signals Interprocess Communication Daemon Processes, shared libraries
› Advanced I/O: Nonblocking I/O, Polling, and Record Locking Encryption
› Network I/O: Sockets, TCP, UDP
› Code reading, coding style, best practices
Grading (Tentative)

› Midterm/Final Exams or Project: 50%
› Programming Assignments: 25%
› Presentations: 25%
Unix history
UNIX history

› Originally developed in 1969 at Bell Labs by Ken Thompson and Dennis Ritchie.

› 1973, Rewritten in C. This made it portable and changed the history of OS.

› 1974: Thompson, Joy, Haley and students at Berkeley develop the Berkeley Software Distribution (BSD) of UNIX.

› two main directions emerge: BSD and what was to become “System V”.

http://www.unix.org/what_is_unix/history_timeline.html
Notable dates in UNIX history

› 1984 4.2BSD released (TCP/IP) 1986 4.3BSD released (NFS)
› 1991 Linus Torvalds starts working on the Linux kernel
› 1993 Settlement of USL vs. BSDi; NetBSD, then FreeBSD are created
› 1994 Single UNIX Specification introduced
› 1995 4.4BSD-Lite Release 2 (last CSRG release); OpenBSD forked off NetBSD
› 2000 Darwin created (derived from NeXT, FreeBSD, NetBSD) 2003 Xen; SELinux
› 2005 Hadoop; DTrace; ZFS; Solaris Containers 2006 AWS ("Cloud Computing" comes full circle)
› 2007 iOS; KVM appears in Linux
› 2008 Android; Solaris open sourced as OpenSolaris
Some UNIX versions

- More UNIX (some generic, some trademark, some just unix-like):

<table>
<thead>
<tr>
<th>1BSD</th>
<th>2BSD</th>
<th>3BSD</th>
<th>4BSD</th>
<th>4.4BSD Lite 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4BSD Lite 2</td>
<td>386 BSD</td>
<td>A/UX</td>
<td>Acorn RISC iX</td>
<td>AIX</td>
</tr>
<tr>
<td>AIX PS/2</td>
<td>AIX/370</td>
<td>AIX/6000</td>
<td>AIX/ESA</td>
<td>AIX/RT</td>
</tr>
<tr>
<td>AMIX</td>
<td>AOS Lite</td>
<td>AOS Reno</td>
<td>ArchBSD</td>
<td>ASV</td>
</tr>
<tr>
<td>Atari Unix</td>
<td>BOS</td>
<td>BRL Unix</td>
<td>BSD Net/1</td>
<td>BSD Net/2</td>
</tr>
<tr>
<td>BSD/386</td>
<td>BSD/OS</td>
<td>CB Unix</td>
<td>Chorus</td>
<td>Chorus/MiX</td>
</tr>
<tr>
<td>Coherent</td>
<td>CTIX</td>
<td>Darwin</td>
<td>Debian GNU/Hurd</td>
<td>DEC OSF/1</td>
</tr>
<tr>
<td>Digital Unix</td>
<td>DragonFly BSD</td>
<td>Dynix</td>
<td>Dynix/ptx</td>
<td>ekkBSD</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>GNU</td>
<td>GNU-Darwin</td>
<td>HPBSD</td>
<td>HP-UX</td>
</tr>
<tr>
<td>HP-UX BLS</td>
<td>IBM AOS</td>
<td>IBM IX/370</td>
<td>Interactive 386/ix</td>
<td>Interactive IS</td>
</tr>
<tr>
<td>IRIX</td>
<td>Linux</td>
<td>Lites</td>
<td>LSX</td>
<td>Mac OS X</td>
</tr>
<tr>
<td>Mac OS X Server</td>
<td>Mach</td>
<td>MERT</td>
<td>MicroBSD</td>
<td>Mini Unix</td>
</tr>
<tr>
<td>Minix</td>
<td>Minix-VMD</td>
<td>MIPS OS</td>
<td>MirBSD</td>
<td>Mk Linux</td>
</tr>
<tr>
<td>Monterey</td>
<td>more/BSD</td>
<td>mt Xinu</td>
<td>MVS/ESA OpenEdition</td>
<td>NetBSD</td>
</tr>
<tr>
<td>NeXTSTEP</td>
<td>NonStop-UX</td>
<td>Open Desktop</td>
<td>Open UNIX</td>
<td>OpenBSD</td>
</tr>
<tr>
<td>OpenServer</td>
<td>OPENSTEP</td>
<td>OS/390 OpenEdition</td>
<td>OS/390 Unix</td>
<td>OSF/1</td>
</tr>
<tr>
<td>PC/IX</td>
<td>Plan 9</td>
<td>PWB</td>
<td>PWB/UNIX</td>
<td>QNX</td>
</tr>
<tr>
<td>QNX RTOS</td>
<td>QNX/Neutrino</td>
<td>QUNIX</td>
<td>ReliantUnix</td>
<td>Rhapsody</td>
</tr>
<tr>
<td>RISC iX</td>
<td>RT</td>
<td>SCO UNIX</td>
<td>SCO UnixWare</td>
<td>SCO Xenix</td>
</tr>
<tr>
<td>SCO Xenix System V/386</td>
<td>Security-Enhanced Linux</td>
<td>SCO UNIX</td>
<td>Sinix ReliantUnix</td>
<td>Solaris</td>
</tr>
<tr>
<td>SPIX</td>
<td>SunOS</td>
<td>Sinix</td>
<td>Trusted Solaris</td>
<td>Ultrix</td>
</tr>
<tr>
<td>Trusted Xenix</td>
<td>TS</td>
<td>Tru64 Unix</td>
<td>Ultrix</td>
<td>Unicox</td>
</tr>
<tr>
<td>Ultrix 32M</td>
<td>Ultrix-11</td>
<td>UCLA Locus</td>
<td>Unicos/mk</td>
<td>Unicox-max</td>
</tr>
<tr>
<td>UNICS</td>
<td>UNIX 32V</td>
<td>Unicos</td>
<td>UNIX System III</td>
<td>UNIX System IV</td>
</tr>
<tr>
<td>UNIX System V/386</td>
<td>UNIX Time-Sharing System</td>
<td>UNIX Interactive</td>
<td>UNSW</td>
<td>USG</td>
</tr>
<tr>
<td>Venix</td>
<td>Wollongong</td>
<td>Xenix OS</td>
<td>Xinu</td>
<td>xMach</td>
</tr>
</tbody>
</table>

2/22/2018 ADVANCED UNIX/LINUX SYSTEM PROGRAMMING 23
UNIX timeline

unix: http://www.levenez.com/unix/
linux: http://futurist.se/gldt/
UNIX everywhere

› Today, your desktop, server, cloud, TV, phone, watch, stereo, car navigation system, thermostat, door lock, etc. all run a Unix-like OS...
UNIX everywhere

› Today, your desktop, server, cloud, TV, phone, watch, stereo, car navigation system, thermostat, door lock, etc. all run a Unix-like OS...

› ...with all the risks that entails.
UNIX basics
UNIX basics: Architecture
System calls and library functions, standards

› System Calls and Library Functions
  – *System calls* are entry points into kernel code where their functions are implemented. Documented in section 2 of the manual (e.g. `write(2)`).
  – *Library calls* are transfers to user code which performs the desired functions. Documented in section 3 of the manual (e.g. `printf(3)`).

› Standards
  – IEEE POSIX (1003.1-2008) / SUSv4
Important ANSI C features, error handling

› Important ANSI C Features:
  – function prototypes generic pointers (void *)
  – abstract data types (e.g. pid_t, size_t)

› Error Handling:
  – Meaningful return values
  – errno variable
  – Look up constant error values via two functions:

```c
#include <string.h>
char *strerror(int errnum)
//Returns : pointer to message string

#include <stdio.h>
void perror(const char *msg)
```
UNIX basics: Pipelines

What is the longest word found on the ten most frequently retrieved English Wikipedia pages?

```bash
for f in $(curl -L http://is.gd/c6F2fs | zgrep -i "^en " | sort -k3 -n | tail -10 | sed -e 's/en \((.*\)) [0-9]* [0-9]/\1/'); do links -d dump http://en.wikipedia.org/wiki/${f}
done |
tr '[:punct:]' '' |
tr '[:space:]' '\n' |
tr '[:upper:]' '[:lower:]' | egrep '^[a-z]+$' |
awk '{ print length() " " $0; }' | sort |
uniq | sort -n | tail -1
```
UNIX basics: Pipelines

Thanks to Douglas McIlroy!

This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

— Douglas McIlroy —
Program design

”Consistency underlies all principles of quality.”
Frederick P. Brooks, Jr
Program design

- UNIX programs...
  - ...are simple
  - ...follow the element of least surprise
  - ...accept input from stdin
  - ...generate output to stdout
  - ...generate meaningful error messages to stderr
  - ...have meaningful exit codes
  - ...have a manual page

Boot/Login process

[...]
total memory = 768 MB avail memory = 732 MB
timecounter: Timecounters tick every 10.000 msec mainbus0 (root)
[...]
boot device: xbd3
root on xbd3a dumps on xbd3b mountroot: trying lfs... mountroot: trying ffs... root file system type: ffs
init: copying out path `/sbin/init' 11 [...]
Starting local daemons:. Starting sendmail.
Starting sshd. Starting snmpd.
Starting cron.
NetBSD/amd64 (panix.netmeister.org) (console)

login: wwyhsu
Password:
Last login: Sat Sep 10 14:27:56 2011 on console
The Regents of the University of California. All rights reserved.
NetBSD 5.0.2 (PANIX-VC) #2: Tue Oct 19 16:30:57 EDT 2010

Welcome to NetBSD!

$
Soooo... what exactly is a "shell"?

$ wget http://course.deepsea9.taipei/linux/simple-shell.c
$ more simple-shell.c
$ cc -Wall -o mysh simple-shell.c
$ ./mysh
$$ /bin/ls [...]$$ ^D
$
Files and directories

- The UNIX filesystem is a tree structure, with all partitions mounted under the root (/).
- File names may consist of any character except / and NUL as pathnames are a sequence of zero or more filenames separated by /’s.
- Directories are special ”files” that contain mappings between inodes and filenames, called directory entries.
- All processes have a current working directory from which all relative paths are specified. (Absolute paths begin with a slash, relative paths do not.)
Listing files in a directory

$ wget http://course.deepsea9.taipei/linux/simple-ls.c
$ more simple-ls.c
$ cc -Wall -o myls simple-ls.c
$ ./myls . [...]  
$
User identification

› **User IDs** and **group IDs** are numeric values used to identify users on the system and grant permissions appropriate to them.

› **Group IDs** come in two types; *primary* and *secondary*.

$ id
Unix time values

› Calendar time: measured in seconds since the UNIX epoch (Jan 1, 00:00:00, 1970, GMT).

› Stored in a variable of type `time_t`.

$ date +%s

Unix time values

- **Process time**: central processor resources used by a process.

- Measured in clock ticks (*clock_t*). Three values:
  - clock time (real time)
  - user CPU time
  - system CPU time

```bash
$ time grep -r _POSIX_SOURCE /usr/include >/dev/null
```
Standard I/O

› Standard I/O:
  
  – **File descriptors**: Small, non-negative integers which identify a file to the kernel. The shell can redirect any file descriptor.
  
  – **Kernel provides unbuffered I/O through e.g.** `open` `read` `write` `lseek` `close`
  
  – **Kernel provides buffered I/O through e.g.** `fopen` `fread` `fwrite` `getc` `putc`

$ wget http://course.deepsea9.taipei/linux/simple-cat.c

$ wget http://course.deepsea9.taipei/linux/simple-cat2.c

$ diff -bu simple-cat*.c [...]

$
Processes

› Programs executing in memory are called processes.
  – Programs are brought into memory via one of the six \texttt{exec(3)} functions.
    › Each process is identified by a guaranteed unique non-negative integer called the \texttt{processes ID}.
    › New processes can only be created via the \texttt{fork(2)} system call.
  – Process control is performed mainly by the \texttt{fork(2)}, \texttt{exec(3)} and \texttt{waitpid(2)} functions.

$ \texttt{wget http://course.deepsea9.taipei/linux/pid.c}$
$ \texttt{more pid.c}$
$ \texttt{cc -Wall -o mypid pid.c}$
$ \texttt{./mypid .}$

[...]
$ \texttt{echo $$}$

[...]
Processes

$ pstree -hapun | more
Processes

 [...] 
 total memory = 768 MB avail memory = 732 MB 
 timecounter: Timecounters tick every 10.000 msec mainbus0 (root) 
 [...] 
 boot device: xbd3 
 root on xbd3a dumps on xbd3b mountroot: trying lfs... 
 mountroot: trying ffs... root file system type: ffs 
 init: copying out path `/sbin/init' 11 
 [...] 
 Starting local daemons:. 
 Starting sendmail. 
 Starting sshd. 
 Starting snmpd. 
 Starting cron. 

NetBSD/amd64 (panix.netmeister.org) (console) 
login: wwyhsu 
Password: 
Last login: Sat Sep 10 14:27:56 2011 on console 
The Regents of the University of California. 
NetBSD 5.0.2 (PANIX-VC) #2: Tue Oct 19 16:30:57 EDT 2010 

Welcome to NetBSD! 

$
Processes
Signals

 Signals notify a process that a condition has occurred. Signals may be:

– Ignored
– Allowed to cause the default action
– Caught and control transferred to a user defined function

$ wget http://course.deepsea9.taipei/linux/simple-shell2.c
$ more simple-shell2.c
$ cc -Wall -o mysh simple-shell2.c
$ ./mysh
$$ /bin/ls
[...]
$$ ^C
Caught SIGINT!
Homework

› read `intro(2)`, Stevens 1 & 2

› Follow, test and understand all examples from this lecture.

› Ensure you have access to a NetBSD/Linux system.