CSE 451 Section Autumn 2005

Richard Dunn
rdunn@cs
Office hours: WTh 3:30-4:20
Allen 216 (or lab)
Reminders

- Sign up for mailing list
- Read the web site
  - Work through lab information
- Start reading the book
- Do the first homework – due Monday!
- Check forkbomb access
- Read & start project 1
451 Projects

■ 4 projects
■ First one – individual, others – groups of 3
■ Need basic C and UNIX skills
  ■ Check links if you need help with this
■ Challenging
  ■ Don’t leave until last minute
■ Learn a lot of cool stuff
First Project

- Introduces C and Unix skills you’ll need
- Teaches how to build and run Linux in VMWare
- Two main parts:
  - Write a simple shell in C
  - Add a simple system call to Linux kernel

- Due: Friday, Oct 7, before lecture (2:30)
  - Electronic turnin: code + writeup
The shell

- Print out prompt
- Accept input
- Parse input
- If built-in command
  - do it directly
- Else create new process
  - Launch specified program there
  - Wait for it to finish
- Repeat
System Calls

- What’s a system call?
- Examples?
- In your shell:
  - Use `fork` to create a child process
  - Use `execvp` to execute a specified program
  - Use `wait` to wait until child process terminates
Project 1: Adding a System Call

- Add `physusage` system call to Linux:
  - Purpose: count memory allocation
  - Make a histogram of allocated memory sizes (powers of 2)

- Steps:
  - Modify kernel to keep track of this information
    - We give you the kernel code
  - Add `physusage` to return the counts to the user
  - Use `physusage` in your shell to get this data from kernel and print it out.
Example of physusage

CSE451Shell% clear_physusage
CSE451Shell% cd /
CSE451Shell% pwd /
CSE451Shell% date
Wed Sep 29 16:52:41 PDT 2004
CSE451Shell% time
       [--portability] [--format=format] [--output=file] [--version]
       [--help] command [arg...]
CSE451Shell% physusage
Total requests to page_alloc: 47
Requests for order 1 pages: 23
Requests for order 2 pages: 20
Requests for order 3 pages: 4
CSE451Shell% exit
Programming in kernel mode

- Your shell will operate in user mode
- Your system call code will be in the Linux kernel, which operates in kernel mode
  - Be careful - different programming rules, conventions, etc.
Programming in kernel mode

- Can’t use application libraries (e.g. libc)
  - E.g. can’t use printf
- Use only functions defined by the kernel
  - E.g. use printk instead
- Include files are different in the kernel
- Don’t forget you’re in kernel space
  - E.g. unsafe to access a pointer from user space directly, use fn’s that perform checks
- Best way to learn – look at existing code
Computing Resources

- Develop your code on dedicated 451 Linux host:
  - forkbomb.cs.washington.edu
- Test your code on VMWare PCs in 006
VMWare

- Software simulation of x86 architecture
- Run an OS in a sandbox
  - Easily reset to known good state
Using VMWare

- All disks are nonpersistent
  - *Powering off loses your changes!* Use “shutdown –r now” instead
- Network adapter is host-only
There is only one user: *root*

The password is *rootpassword*

You will need to:

- Build a kernel image on *forkbomb*
- Transfer it to Linux running inside VMWare
- Boot your new Linux kernel in VMWare

Use ftp to get your files into VMWare

- FTP to 192.168.93.2 from the host running VMWare.
  - E.g. using IE, go to ftp://root:rootpassword@192.168.93.2
UNIX & C help

- Unix & C tutorial links on 451 projects page
- What if my shell crashes?
  - Use gdb to debug
  - gdb tutorials linked on web site
- What do I use to compile my shell?
  - gcc. For example, `gcc -o shell shell.c -Wall -g`
- What do I use to type up my code?
  - I recommend Emacs (look for Emacs tutorials)
  - VS.NET works too
UNIX & C help - 2

- How do I find stuff in the kernel source?
  - Use `grep -r search_string *`
  - Use LXR (Linux Cross Reference): [http://lxr.linux.no/](http://lxr.linux.no/)

- Which library functions does C have to simplify my shell code?
  - `man strncmp, gets, fgets, strtok, strchr, perror`
Refreshing C skills; code quality

- What’s wrong with this:
  ```c
  char *buffer;
  buffer = malloc(100);
  strcpy(buffer, param);
  ```

- How do we fix this?
C Debugging hint

#define MYDEBUG // comment out to disable debugging

#ifdef MYDEBUG
    #define DEBUG(x) x
#else
    #define DEBUG(x)
#endif

...

int main() {
    ...
    printf("normal output");
    DEBUG(printf("debug output"));
    ...
}
More debugging

- Just for printing:

```c
#ifdef MYDEBUG
    #ifdef __KERNEL__
        /* This one if debugging is on, and kernel space */
        # define PDEBUG(fmt, args...) printk("myprg: " fmt, ## args)
    # else
        /* This one for user space */
        # define PDEBUG(fmt, args...) fprintf(stderr, fmt, ## args)
    # endif
#else
    # define PDEBUG(fmt, args...) /* not debugging: nothing */
#endif
```