CSE 451 Section
Autumn 2004

Alex Moshchuk
anm@cs
Office hours: Tue 1-2, Thu 4:30-5:30
Allen 218 (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 tomorrow!
- Check spinlock/coredump 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, Jan 14, 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 `execcounts` system call to Linux:
- Purpose: collect statistics
- Count number of times you call `fork`, `vfork`, `clone`, and `exec` system calls.

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

Example of execcounts

```
CHE451Shell> execcounts clear
CHE451Shell> execcounts
CHE451Shell> date
Wed Sep 29 1:57:41 PDT 2004
CHE451Shell> lstat
CHE451Shell> execcounts
Statistics:               -------
Fork:                    3       27%
Clone:                   0       0%
VFork:                   0       0%
Exec:                    8       72%
CHE451Shell> 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 hosts:
  - `spinlock`, `coredump`
- Test your code on VMWare PCs in 006
- Do not use attu

VMWare

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

- Power on/off, reset
  - [Image of VMWare setting]
- All disks are nonpersistent
  - Powering off loses your changes! Use "shutdown -r now" instead
- Network adapter is host-only

Linux & VMWare

- There is only one user: root
- The password is rootpassword
- You will need to:
  - Build a kernel image on spinlock/coredump
  - 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/
- Which library functions does C have to simplify my shell code?
  - man strcmp, gets, fgets, strtok, strchr, perror

Refreshing C skills; code quality

- What’s wrong with this:
  - 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 _KERNEL_
    /* This one if debugging is on, and kernel space */
    define PDEBUG(fmt, args...) printk(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
```

works for both for kernel and userspace

To use:

```c
PDEBUG("Testing two numbers: %d and %d\n", num, num2);
```