

## CSE 451 Section Autumn 2004

Alex Moshchuk  
anm@cs  
Office hours: Tue 1-2, Thu 4:30-5:30  
Allen 218 (or lab)

1

## Reminders

- n Sign up for mailing list
- n Read the web site
  - n Work through [lab information](#)
- n Start reading the book
- n Do the first homework – due tomorrow!
- n Check spinlock/coredump access
- n Read & start project 1

2

## 451 Projects

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

3

## First Project

- n Introduces C and Unix skills you'll need
- n Teaches how to build and run Linux in VMWare
- n Two main parts:
  - n Write a simple shell in C
  - n Add a simple system call to Linux kernel
- n Due: Friday, Jan 14, before lecture (2:30)
  - n Electronic turnin: code + writeup

4

## The shell

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

```
CSE451shell% /bin/date
Fri Jan 16 00:05:39 PST 2004
CSE451shell% pwd
/root
CSE451shell% cd /
CSE451shell% pwd
/
CSE451shell% exit
```

5

## System Calls

- n What's a system call?
- n Examples?
- n In your shell:
  - n Use *fork* to create a child process
  - n Use *execvp* to execute a specified program
  - n Use *wait* to wait until child process terminates

6

## Project 1: Adding a System Call

- n Add *execcounts* system call to Linux:
  - n Purpose: collect statistics
  - n Count number of times you call *fork*, *vfork*, *clone*, and *exec* system calls.
- n Steps:
  - n Modify kernel to keep track of this information
    - n We give you the kernel code
  - n Add *execcounts* to return the counts to the user
  - n Use *execcounts* in your shell to get this data from kernel and print it out.

7

## Example of execcounts

```

CSE451Shell% execcounts clear
CSE451Shell% cd /
CSE451Shell% pwd
/
CSE451Shell% date
Wed Sep 29 16:52:41 PDT 2004
CSE451Shell% time
Usage: time [-apvV] [-f format] [-o file] [--append] [--
verbose]
        [--portability] [--format=format] [--output=file] [--
version]
        [--help] command [arg...]
CSE451Shell% execcounts
Statistics:
Fork:          3      27%
Clone:        0       0%
VFork:        0       0%
Exec:         8      72%
CSE451Shell% exit
  
```

8

## Programming in kernel mode

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

9

## Programming in kernel mode

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

10

## Computing Resources

- n Develop your code on dedicated 451 Linux hosts:
  - n *spinlock*, *coredump*
- n Test your code on VMWare PCs in 006
- n Do not use attu

11

## VMWare

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



12

## Using VMWare

Power on/off, reset

VMWare config  
Don't change!

- All disks are nonpersistent
  - Powering off loses your changes! Use "shutdown -r now" instead
- Network adapter is host-only

13

## 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`

14

## 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

15

## 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 `strncmp`, `gets`, `fgets`, `strtok`, `strchr`, `perror`

16

## Refreshing C skills; code quality

- What's wrong with this:
 

```
char *buffer;
buffer = malloc(100);
strcpy(buffer, param);
```
- How do we fix this?

17

## 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"));
    ...
}
```

18



## More debugging

- n Just for printing:

```
#ifndef 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
```
- n works for both for kernel and userspace
- n To use:

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

19