Reminders

- Project 1 due tomorrow by 6:00
- Office hours in 006 today at 4:30
- Start thinking about project groups (3 people) for the rest of the quarter
  - Groups due by Tuesday noon (email anm@cs)
  - After that we'll pick a group for you
- Today: project 1 questions

1

Project 1 – issues

- C strings
- Copy_to/from_user and counters
- Syscalls: macros ; arguments
- Execvp, wait
- Other things??

2

C strings

- You only need to use:
  - **strncmp**(src,dest,256) – compare strings, 0 if equal, not 0 o.w. *Do not do str1 == str2!!!*
  - **strtok**:
    - 1st use: tok = strtok(buf, "*delimiters*");
    - Subsequent uses: tok = strtok(NULL, "*delimiters*");
  - **fgets**(buf, 256, stdin) – read a line (up to 256 chars) from stdin (or getline)
  - *(maybe)* **strncpy**(dest, src, 256) – copy up to 256 chars from src to dest.
  - *(maybe)* Allocate memory with **malloc**, free with **free**
- Fine to assume:
  - A maximum length for a shell command (say, 256)
  - Maximum number of arguments (say, 256 again)

3

Passing counters

- Do not printk the statistics in execcounts!!!
- Execcounts should pass count values to the shell
  - The shell then prints out statistics
- Copying counters to userspace:
  - Shell passes in something to hold data
  - sys_execcounts fills the data in

4

Copying data to/from kernel

- Unsafe to directly access user pointers!

```
long sys_gettimeofday(struct timeval *tv)
{
    if (tv) {
          struct timeval ktv;
          do_gettimeofday(&ktv);
          if (copy_to_user(tv, &ktv, sizeof(ktv)))
                return -EFAULT;
    }
    return 0;
}
```

- copy_to/from_user return amount of uncopied data

5

Syscalls

- Two ways to use one:
  - Linux style:
    - in <asm/unistd.h>:

```
#define __NR_foo 292
static inline _syscall2(int, foo, int, arg1,
                                char *, arg2)
```

    - In userspace, just call `foo(4,"test");`
  - BSD style:
    - in shell.c:

```
#define __NR_foo 292
ret = syscall(__nr_foo, arg1, arg2);
```

6

How syscalls work

```
In entry.S:

ENTRY(system_call)
        pushl %eax                  # save orig_eax
        SAVE_ALL
        cmpl $(NR_syscalls),%eax
        jae badsys
        call *SYMBOL_NAME(sys_call_table)(,%eax,4)
        movl %eax,EAX(%esp)         # save the return value
restore_all:
        RESTORE_ALL
```

7

Execvp

- execvp:
  - You must build an array of strings to pass to it
  - Make sure the last thing in this array is NULL
  - Make sure the array includes the program name

8

Wait

- wait(int *status)
  - "man 2 wait" get information about it.
    - "man wait" by default goes to the shell reference!
  - What's wrong with this code:

    int *status;

    wait(status);

    Fix??

9

Extern

- How do we access global variables defined in one file from another file?

10

Other things

- Check that every malloc has a matching free
- Check for all errors
  - E.g. malloc returns NULL
  - Frequently, global constant errno will be set
  - Use *perror("error description");* to see what the error was.
- Don't worry about architectures other than x86.
- Don't worry about compiling the shell in vmware
  - Compile on spinlock, transfer executable to vmware
- Q: "warning: implicit declaration of xyz" -- ???
  - A: Check include files

11