### 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 ann@cs)
  - After that we’ll pick a group for you
- Today: project 1 questions

### Project 1 – issues
- C strings
- Copy_to/from_user and counters
- Syscalls: macros; arguments
- Execvp, wait
- Other things??

### C strings
- You only need to use:
  - `strcmp(src, dest, 256)` – compare strings, 0 if equal, not 0 o.w.  `Do not do str1 == str2!!!`
  - `strtok`:
    - 1st use: `tok = strtok(buf, “delimiter”);`
    - Subsequent uses: `tok = strtok(NULL, “delimiter”);`
  - `fgets(buf, 256, stdin)` – read a line (up to 256 chars) from stdin (or getline)
  - `(maybe) `strcpy(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)

### Passing counters
- Do not print the statistics in execounts!!!
- Execounts 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_execounts fills the data in

### 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`

### Syscalls
- Two ways to use one:
  - Linux style:
    - in <asm/unistd.h>:
      - `#define __NR_foo 232`
      - `static inline __syscall2(int, foo, int, arg1, char *, arg2)`
        - In userspace, just call `foo(4,”test”);`
  - BSD style:
    - in shell.c:
      - `#define __NR_foo 232`
      - `ret = syscall(__nr_foo, arg1, arg2);`
**How syscalls work**

In entry S:

ENTRY(system_call)
  push %eax
  save orig %eax
cmpl %EBX, %eax
jne badsys
-call "SYMBOL_NAME(sys_call_table)"(%eax,4)
movl %eax, %AX(%esp)
  # save the return value
restore_all:
  RESTORE_ALL

---

**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

---

**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??

---

**Extern**

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

---

**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