Operating System

- Service provider to "users"
  - execute (run) programs (Thread control)
- Input/Output
  - local machine (serial, USB, disk, ...)
  - network (eg. ether/wifi and network stack)
- Memory Management
- File Management

- May not provide User Interface
  - X Windows - not part of OS
  - Mac OS X user interface not part of the OS
  - shell - not part of OS
    - interactive -- run commands
    - shell scripts
      - many UNIX commands for scripts
  - built-in vs programs
- Win 32 -- originally not in OS
  - later moved into OS for speed
System Calls vs. Library Calls

- **System Call**
  - Programmed request to Operating system
  - API looks like C or some language
    - `write (fd, mystring, nbytes)`
    - `fork ()`
    - `NtCreateFile(...), NtReadFile(...) (Windows)`

- **Library Call**
  - A utility "user land" function / no OS computation
    - `atoi (char *)`
    - `qsort(base, nmemb, size, compare)`
    - `sqrt(value) (libm, -lm to link)`
  - Some library functions use systems call to do job
    - `printf ("%s", mystring);`
    - `newwin(lines,cold,begin_y,begin_x) (libcurses, -lcurses)`

- `ant.c`
Manual pages

- man section 1 -- User Commands
  - man ls
  - man intro
  - man man
  - man write
  - man printf
  - man sh
  - man csh
  - man mkdir
  - man cp
  - man rm
  - man vi
Manual pages (page 2)

- man section 2 -- OS calls
  - man 2 open
  - man 2 intro
  - man 2 write
  - man 2 chdir

- man section 3 -- Library calls
  - man 3 fopen
  - man 3 intro
  - man 3 printf
  - man 3 getenv
An example shell -- minishell.c

- **Library Calls**
  - `fprintf(3)` - C stdio
  - `fgets(3)` - C stdio
  - `feof(3)` - C stdio
  - `perror(3)` - C stdio
  - `strlen(3)` - String library

- **System Calls**
  - `fork(2)` - Duplicate the process
  - `wait(2)` - Wait for a process to exit

- **Library wrappers for system calls**
  - `execlp(3)` - start a program