System Call Programming & Debugging

Week 6

Processor Modes

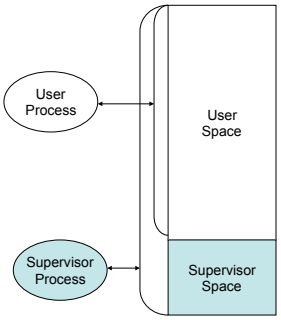
- Mode bit used to distinguish between execution on behalf of OS & execution on behalf of user.
- Supervisor mode: processor executes every instruction in its hardware repertoire.
- User mode: can only use subset of instructions

Processor Modes

- Instructions that can be executes in supervisor mode are supervisor, privileges, or protection instruction
  - I/O instructions are protected. If an application needs to do I/O, it needs to get the OS to do it on its behalf.
  - Instructions that can change the protection state of the system are privileges (e.g., process' authorization status, pointers to resources, etc)

Processor Modes

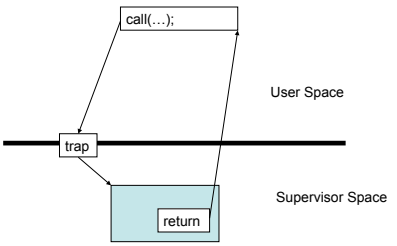
- Mode bit may define areas of memory to be used when the processor is in supervisor mode vs user mode



The Kernel

- Part of system software executing in supervisor state
- Trusted software: implements protection mechanisms that could not be changed through actions of untrusted software in user space

System Calls



Example System Calls

- #include<unistd.h>
- pid_t **getpid**(void);
  - Returns the process ID of the calling process
- int **dup**(int fd);
  - Duplicates a file descriptor fd. Returns a second file descriptor that points to the same file table entry as fd does.
- **fstat**(int filedes, struct stat *buf)
  - Returns information about the file with the descriptor filedes into buf
- Why are these system calls and not just regular library functions?

More System Calls

- ssize_t **read**(int fildes, void *buf, size_t nbyte)
  - fildes: file descriptor
  - buf: buffer to write to
  - nbyte: number of bytes to read
- ssize_t **write**(int *fildes*, const void **buf*, size_t *nbyte*);
  - fildes: file descriptor
  - buf: buffer to write from
  - nbyte: number of bytes to write

Laboratory

- Write a program using **getchar** and **putchar** to copy all bytes in stdin to stdout
- Write a program that uses **read** and **write** to read and write each byte, instead of using getchar and putchar. The nbyte argument should be 1 so it reads/writes a single byte at a time.
- Test it with bigfile.txt from http://www.cs.ucla.edu/~rpon/bigfile.txt

Laboratory

- read and write are system calls.
- getchar and putchar are library functions that live in user space but call read and write. And do buffering!

Buffering Issues

- What is buffering?
- Why do we buffer?
- Can we make our buffer really big?

strace and time

- **strace**: Intercepts and prints out system calls to stderr or to an output file.

  strace -o strace_output ./catb < bigfile.txt
- **time**: does timing… duh

  time ./catb < bigfile.txt

Buffered versus Unbuffered

- How many read and write calls are there in each?
- Why does one take longer than the other?

Writing to a File vs. Terminal

- Do you see any differences in the read/ writes?
- In the buffered version, how many bytes are being read/written when outputting to a file and when outputting to the terminal?
- Why is there a difference?
- What are the performance considerations?

Homework

- Rewrite binsort using system calls
- If stdin is a regular file, have your program initially allocate enough memory to hold all data in the file all at once.
- Otherwise, it should perform the same way
- You must also output, to stderr, how many comparisons you made
- In addition to the functions you used last time, you'll need **read**, **write**, and **fstat**
- Read the man pages for these functions
- The program should be named **binsortu**
- Measure differences in performance between **binsortr** and **binsortu** using the **time** command.