CSci 4061

Introduction to Operating Systems

OS Concepts and Structure
The Kernel: core layer of the OS

• The kernel is a library of procedures shared by all user programs, but the kernel is protected:
  – User code cannot access internal kernel data structures directly
  – User code can invoke the kernel only at well-defined entry points, and these are?

• Kernel code is like user code, but the kernel is privileged:
  – Kernel has direct access to all hardware, and handles interrupts and hardware exceptions
  – CPU is either executing OS code (kernel-mode) or your code (user-mode)

OS can be a mix of user-mode and kernel-mode
Systems Programmer Viewpoint

- Systems programmer can use system calls directly
  - executed by the OS (i.e. kernel mode)
  - when efficiency demands it
  - assembly code: x86 “int” instruction, e.g. int 48

- Alternatively, language-specific libraries can be used to access system calls
  - C programming language libraries (libc.a)
  - E.g. read (…)
Terminology Alert!

• I will often refer to low-level library calls as system calls
  – e.g. `read (...)``;
  – becomes `int x`

• Library (or system calls) are not part of the C language
Running programs: memory and the CPU

Program B makes a system call
Let’s Look At

OS Concepts and Abstractions Above the Hardware
Abstraction

High-level construct

Useful, easy-to-use, understand

Hides lower-level details

PL: class or structure data-type
Operating System Concepts: Process

Process is an executing program: container for computing resources (abstraction)

- Process tree
- A created two child processes, B and C
- B created three child processes, D, E, and F

What resources?
A thread is an executing stream of instructions normally within a process

- A has two threads; share A’s resources
- Every process has at least one thread
- Threads can also exist in the OS

```c
main () {
    int i;
    i=2;
}
```
Operating System Concepts: Synchronization

• Concurrency (processes/threads run together) and shared resources can lead to problems:
  – (a) Race condition
  – (b) Deadlock

• Solution: Synchronization, e.g. case a?
Operating System Concepts: Synchronization Issues

Livelock! (aka “Minnesota Nice”)  
No one makes progress

Deadlock is often caused by poor use of synchronization
Files/directories are an OS abstraction to make data storing and sharing easier.
Operating System Concepts: Communication

- Two processes connected by a “pipe”, channel
- Processes need to communicate - why?
Operating System Concepts: Memory Management

• How is memory allocated to programs?
  – Largely an “inside” issue but ....
  – We will see how a program can make good use/bad use of memory

• Abstraction = virtual memory
System Calls

System calls are the way user programs interact with the OS

- Generally available as assembly-language instructions
- C-Unix provides a library interface to system calls to avoid this messiness
- e.g. `read (...)` gets compiled into the appropriate syscall linkage/assembly code
- Difference?
In this course, we will use the term system call to refer to the C-Unix interface, e.g. `open`
Systems Concepts

“systems”: OS, Internet, ATC, ...

Granularity
Modularity
Abstraction
Layering
Hierarchy
Complexity
Complexity?

• Different stakeholders => different metrics and requirements
  – Programmer => ease-of-problem-solving
  – End-user(s) => performance, ease-of-use
  – Owner (~ system) => fairness/priority, efficiency or utilization
  – Admin => security
  – OS Vendor => extensible, secure, reliable, ...

• Tradeoff and conflict lead to complexity
Next Time

Programs and Processes in C and UNIX

Read Chapter 2,3 (R&R), opt: Chapter 2 (MOS) or Chapter 3 (S&G)

Have a great weekend

Recitation on Monday
This Weekend

• C/UNIX Refresh (or cram)

1. Edit and write a simple C program
2. Compile and run it
3. Look at a debugger such as DDD, GDB