Introduction (CSE 422S)

Ken Wong
Washington University

kenw@wustl.edu
www.arl.wustl.edu/~kenw

Miscellaneous (1)

See Web page for general information
» Follow link from: http://www.arl.wustl.edu/~kenw
» Books
  • Required: Andrew Tanenbaum, "Modern Operating Systems, Second Edition"
» Grading (Not Curved)
  • 40% Exams (2)
  • 40% Labs/Projects (2) - Programs in C/C++
  • 20% Homework (8-10), quizzes and miscellaneous
    » Reading quizzes (graded)
    » Concept quizzes (NOT graded, provide feedback)

Miscellaneous (2)

Prerequisites: CS 342S/CSE 332S (OO Software)
  » Concepts: Abstraction, class, instance, interface, semantics
    • e.g., Objects have state and respond to requests (messages). Some requests return state while others modify state
  » Concrete Language: C++
    • "Glorified C", "Dirty Java"
    • Just another imperative programming language
    • We use: Formatted I/O (e.g., printf), simple classes, some templates (e.g. map)

Computer Accounts
  » CEC Unix systems, Sever 214
  » We use the CEC Linux machines
    • e.g., hive.cec.wustl.edu

CSE 422S

What it is NOT about
» Programming languages, Kernel programming, OS survey

Lectures
  » "It’s impossible to learn much by simply sitting in lectures ..." (Richard Feynman)
  » Elaborate on reading
  » Address potential difficulties

Homework exercises
  » Deepen understanding
  » Build your confidence

Projects
  » Integrate knowledge ... Reflect on concepts

Understanding is not just Remembering !!!
An Operating System

- **OS**: A collection of system programs which allow the user to run applications
  - Provides user with an **abstract machine**
    - Abstraction is used to handle complexity
  - Provides **virtual** (logical) resources
    - Virtualization supports resource sharing
  - Provides mechanisms for managing resources
    - Resource management aims to provide good quality of service

- **Properties of a good OS**
  - Easy to use
  - Isolation/Protection from other users’ errors
  - Efficient and fair use of resources
  - Reliable operation
  - Secure from unauthorized users

---

Operating System Facilities

- **Processes**
  - Resource containers that can be **controlled** and **scheduled**

- **Filesystem and I/O**
  - Structured and unstructured **byte streams**
  - Access through **system calls**

- **Virtual Memory**
  - Logical address space **mapped** to actual address space

- **Interprocess Communication and Synchronization**
  - Communicating multiple threads of control

- **Timers**
  - Supports scheduling and accounting

- **Initialization**
Programs and Processes

```
#include <unistd.h>
#include <stdio.h>
int main (int argc, char *argv[]) {
    printf("I'm process %d with %d parameters\n", getpid(), argc);
    exit(0);
}
```

- **Program**: An executable file residing on disk
  - A partial machine image
- **Process**: An executing instance of a program
  - Has a run-time state

```
hilton> gcc -o foo hello.c
hilton> foo
I'm process 7164 with 1 parameters
```

Process Facility

- **Abstraction**
  - **Resource Consumer** (memory, CPU, files)
  - **Execution Thread** (1 program counter)
  - Attributes: Identifier, accounting info, access rights
  - Operations: create, kill, stop, continue, clone, send, recv
- **Virtualization**
  - **Dedicated** virtual address space vs. **Shared** physical address space
  - **Dedicated** virtual CPU vs. **Shared** actual CPU
- **Resource Management**
  - Multiplex process resource demands for high aggregate throughput and low individual response time
  - What are the policies? How are they enforced?

A Contention Problem

- IF:
  - Resources can not be shared
  - A process will require two resources before proceeding
- How can we always ensure forward progress?
- How can we also ensure high speed?
- Can we generalize?

Interprocess Synchronization Facility

- **Abstraction**
  - Two processes can share their address spaces
    - A process can read from (write to) a variable that appears in both address spaces
- **Virtualization**
  - **Actual**: Contents of memory location can appear in multiple hardware places
    - e.g., Main memory, store buffers, cache memories, registers
  - **Virtual**: Shared variable is updated **atomically**
- **Resource Management**
  - Virtualization typically not supplied by OS
  - OS may supply primitives for implementing atomic update
The Shared-Resource Problem

IF
- The activity of each car is viewed as a process
- Only a "lead" car can switch the go/no-go signs

Design an algorithm that ensures
- No more than one car can be on the bridge
- Fair and maximum usage of the bridge

A Communication Problem

Operation
- Client sends request to server
- Server handles request and responds with reply
- OS sends async request to process

Virtual Memory Facility

Abstraction
- Finite linear address space (virtual memory)
- Nearly infinite potential size

Virtualization
- Dedicated logical address space; Shared physical address space

Resource Management
- Physical memory allocation (How large a page?)
- Load control (#Pages to allocate to each process?)
- Replacement strategy (Swap which page to disk?)
- Physical memory sharing between processes (How?)
Virtual Memory

Note unmapped pages.

Page Allocation/Replacement Problem

- Let R represent the sequence of page numbers referenced (read/write) by a process
  - Each element of R is a non-negative integer from 0 to P-1
  - R is “representative” of all processes
- Devise a page replacement algorithm that will minimize the processes’ run-time
  - If you allocate F page frames to each process
  - Variation: If the number of page frames can vary over time for each process
  - How can we efficiently generate representative reference sequences?

File System Facility

- Abstraction
  - A byte stream but physical storage is highly structured
  - Operations (unbuffered): open, seek (move), read, write, close
  - Operations (buffered): fopen, fseek (move), fread, fwrite, fclose
- Virtualization
  - Dedicated user buffers: Shared kernel buffers
  - Dedicated file channel: Shared I/O bus
- Resource Management
  - Disk space quota (How much space?)
  - Access control (Who owns? Can read/write?)
  - Scheduling policy (When?)

Disk Storage Recovery Problem

- Design a high-availability disk storage system
  - Pr[R or W to drive A and B fails] ≈ 0 (i.e., negligible)
  - Pr[R or W to drive A fails], Pr[R or W to drive B fails] > 0
- Design an algorithm that provides high disk storage availability
  - Use standard disk controllers
  - Pr[R or W fails] ≈ 0
  - Can handle: 1) Media failure, and 2) System crash
- What algorithms can be employed if you are allowed to design a custom disk controller?
  - What will be the cost of this solution?
To Do 1

- Send me email in the following format:
  
  » Put "cse422s" in the Subject line
  » Put the information in the **BODY** of the email as **plain text**
  » Do NOT mime-encode the line
  » Do NOT send as an attachment

  
  YOUR Name (lower case, last name first, no spaces)

  cse422s  wong.ken  kenw@arl.wustl.edu

To Do 2

- Read Tanenbaum, Chapter 1
  
  » Focus on
    
    • Section 1.4 Computer Hardware Review
    • Section 1.6 System Calls

- Read [An Introduction to Linux, Machtelt Garrels](www.tldp.org/guides.html)