Chapter 6

Process Management

OS organization

Process Management Tasks

- Define & implement the essential characteristics of a process and thread
  - Algorithms to define the behavior
  - Data structures to preserve the state of the execution
- Define what "things" threads in the process can reference – the address space (most of the "things" are memory locations)
- Manage the resources used by the processes/threads
- Tools to create/destroy/manipulate processes & threads

Process management (...ctd)

- Tools to time-multiplex the CPU – Scheduling the (Chapter 7)
- Tools to allow threads to synchronize the operation with one another (Chapters 8-9)
- Mechanisms to handle deadlock (Chapter 10)

Introduction

- Scenario
  - One process running
  - One/more process performing I/O
  - One/more process waiting on resources

  Most of the complexity stems from the need to manage multiple processes
**Process Manager Overview**

- **Program**
- **Process**

**Abstract Computing Environment**

- **File Manager**
- **Device Manager**
- **Memory Manager**
- **Scheduler**
- **Resource Manager**

**Process Description**

- **Deadlock Protection**
- **Synchronization**

**Devices**

**Memory**

**CPU**

**Other H/W**

**Process components**

- **Program** defines behavior
- **Data**
- **Resources**
- **Process Descriptor** keeps track of process during execution

**Process Descriptor**

- **Process**
  - **Owner**
  - **Parent**
  - **Children**
  - **Siblings**
  - **Siblings**
  - **Address Space**
    - **Definition**
     - **Program**
     - **Variables**
     - **...**
     - **Address space is generated/defined by translation**

**Creating an Executable Program**

- Separate objects each relative to 0
- One large program
- Y = (X + Y)
- Maps relative address space to physical memory location
- Generates separate object code modules

- Relocates modules one behind other
- Relocates addresses of all but first
- Relocates external reference to library calls and external modules

**Basic Memory Hierarchy**

- Fastest
- Secondary memory, $M_2$
- Primary memory, $M_1$
- Access Speed
- Slowest

**Cache memory**
Basic Memory Hierarchy...
- At any point in the same program, element can be in
  - Secondary memory $M_s$
  - Primary memory $M_p$
  - Registers $M_r$

Consistency is a Problem
- $M_s = M_r = M_p$ (code vs data)
- When does one make them consistent?
- How?

Consistency Problem...
- Is Data a Problem ???
  - YES
  - Variable temporarily stored in register has value added to it
  - Therefore, $M_r = M_p$

- On context switch, all registers are saved
  - Therefore, current state is saved

Consistency Problem
- Scheduler switching out processes – Context Switch
- Is Instruction a Problem ???
  - NO
  - Instructions are never modified
  - Separate Instruction and Data space
  - Therefore, $M_i = M_p = M_r$

Sample Scenario...
- Suppose "MOV X Y" instruction is executed
  - $M_y = M_p$

- On context switch, is all of a process’ memory flushed to $M_r$?
  - No, only on page swap

- Hence, $env_{process} = (M_r + M_p) + (...)$

  Note:
  - Flushing of memory frees it up for incoming process
    => Page Swap

Process States
- Focus on Resource Management & Process Management

Recall also that part of the process environment is its *state*

State Transition Diagram

Process States...
- When process enters 'Ready' state, it must
  compete for CPU. Memory has already
  been allocated

- Process has CPU

- Process requests resource that is
  immediately available \( \Rightarrow \) NO blocking

- Process requests resource that is NOT yet
  available

- Resource allocated, memory re-allocated?

State Transition Diagram
Resources & Resource Manager

- 2 types of Resources
  - Reusable (Memory)
  - Consumable (Input/Time slice)

![Diagram of resource management](image)

Process Hierarchy

- Conceptually, this is the way in which we would like to view it
- Root controls all processes i.e. Parent

Creating Processes

- Parent Process needs ability to
  - Block child
  - Activate child
  - Destroy child
  - Allocate resources to child

- True for User processes spawning child
- True for OS spawning `init`, `getty`, etc.
- Process hierarchy a natural, if `fork/exec` commands exist

Factoring in additional Control Complexities

- Recall:
  - A parent process can `suspend` a child process

- Therefore, if a child is in `run` state and goes to ready (time slice up), and the parent runs and decides to suspend the child, then how do we reflect this in the process state diagram ???

- We need 2 more states
  - Ready suspended
  - Blocked suspended

Process State diagram reflecting Control

- Not Blocked
- Suspended
- No memory
Processes in Linux

- Also called tasks
- Task table or process table defined in src/linux/include/sched.h
  ```c
  extern struct task_struct *pidhash[PIDHASH_SZ];
  ```
- Can also be accessed as a doubly-linked list `p->next_task` and `p->prev_task`

Process States

- Linux identifies following states
  1. TASK_RUNNING
  2. TASK_INTERRUPTIBLE
  3. TASK_UNINTERRUPTIBLE
  4. TASK_ZOMBIE
  5. TASK_STOPPED
  6. TASK_EXCLUSIVE

Process Creation

- Remember in traditional UNIX, we use `fork()` and then typically `exec()`.
- `fork()` duplicates resources owned by parent for child process and copies them to new address space.
- This method is slow and inefficient, since `exec()` wipes out address space anyway.
Process creation in Linux

- Copy On Write technique
- Lightweight processes
- vfork()

Copy-on-write

- Child pages are pointers to parent pages
- If child makes a change to a page, a new copy is made for the child
- This way, you avoid making separate copies of pages unnecessarily

Lightweight processes

- Allow parent and child processes to share many kernel data structures
- Created in Linux by function called __clone()
- Uses non-standard clone() system call

vfork()

- Creates a process that shares memory address of parent
- Parent is blocked until child exits or executes a new program by doing exec()

User view of processes

- Can use ps command with various options, for example,
  - ps -aux
  - ps -ef

/proc file system

- Process information pseudo file system
- Do man proc to get more info
- /proc directory contains
  - Numerical subdirectory for each running process
  - A number of other files containing kernel table information
/proc... continued

- Files include
  - cpuinfo – contains CPU specs
  - uptime – time in secs since machine was last rebooted and idle time since then
  - version – kernel version
  - loadavg – Load average of machine over the past 1, 5 and 15 minutes
  - ...

Process directories

- One subdirectory for each running process
- Files include
  - cmdline
  - cwd
  - environ
  - exe
  - fdm
  - map
  - mem
  - root

References

- Linux Kernel 2.4 internals, Tigran Aivazian http://www.tldp.org/LDP/lki/
- Modern Operating Systems, 2nd Ed., A. Tanenbaum
- Understanding the Linux Kernel, D. Bovet, and M. Cesati