


CSE3320, Operating Systems

Spring 2020, Homework 1

Please read this:

- 1) Homework assignments are to be completed individually, no teamwork.
- 2) Total points possible: 100 pts.
- 3) If not otherwise stated, each problem is 5 pts.
- 4) Please add the following statement in the beginning of your submission.

 I have neither given or received unauthorized assistance on this work

Signed:

Date:

Chapter 1:

1. What are the two main functions of an operating system? Explain each of the function briefly.

The two main functions of an OS are hardware abstraction and resource management.

As the abstraction, an OS hides messy details of the underlying hardware devices and presents users with a consistent, easy-to-use, and reliable interface. With such an abstraction, an OS is also able to form protection domains among users.

As the resource manager, an OS shares resources among users, programs by multiplexing resources in both time and space.

2. What are the differences between a trap and an interrupt?

Traps are initiated by programs; interrupts are triggered by hardware devices.

Traps are synchronous; interrupts are asynchronous.

3. What are the main structures of contemporary operating systems, and their advantages and disadvantages?

Monolithic kernel is a single large processes running entirely in a single address space. It is a single static binary file. All kernel services exist and execute in kernel address space. The kernel can invoke functions directly.

Advantages:

1. Faster
2. Easier to implement

Disadvantages:

1. Huge kernel, harder to maintain
2. No protection between kernel components
3. Complex dependencies among components, not easily extensible

In Microkernels, the kernel is broken down into separate processes, known as servers. Some of the servers run in kernel space and some run in user-space. All servers are kept separate and run in different address spaces. The communication in microkernels is done via message passing. The servers communicate through IPC (Interprocess Communication).

Advantages:

- 1. Modular design, easily extensible*
- 2. Easy to maintain*
- 3. More reliable and secure*

Disadvantages:

- 1. Performance loss due to message passing-based communication*
- 2. Process management, such as accounting, scheduling could be complicated*

4. How can I/O devices notify an OS of the completion of jobs? List three ways.
Polling, interrupt, and DMA.

5. Describe the actions an OS must take to process an interrupt.

- 1. Enable the interrupt by writing into the corresponding device registers.*
- 2. The hardware device signals the interrupt controller.*
- 3. Interrupt controller informs a CPU.*
- 4. The CPU accepts the interrupt and calls the corresponding service routine.*

6. What is a system call? How is the transition between the user-mode and the kernel-mode performed?

A System call is the interface exposed by the OS to user programs. It provides a way for user programs to use kernel services. When a program triggers a system call, it first pushes its parameters and the return address onto the stack. Then, the TRAP instruction is executed switching from user-mode to kernel-mode, from where kernel services are started. When the work is done, control is returned to the user-mode.

Chapter 2:

1. What is a process? What are the two essential parts of a process? How is a process different from a program?

A process is an abstraction of a running program. It has two essential parts: the sequential execution of instructions and the process state.

A program is represented by static code and data, while a process is a dynamic instantiation of a program.

2. Given the five-state process model, explain how does a process transit among these states and on what events?

Process creation: → new

Process admission: new → ready

Scheduler dispatch: ready → running

Timer interrupt or forced preemption: running → ready

I/O or event wait: running → blocked

I/O or event completion: blocked → ready

Exit or killed: running → terminated

3. What are the differences of threads and processes?

Processes have their own address spaces. Threads share address spaces including shared code, data, heap, and files. Threads are inexpensive to create and to do context switch. However, processes usually have better isolation and protection from other processes.

4. What is multiprogramming and why it is needed?

Multiprogramming is the rapid switching between processes giving the illusion of running multiple programs in parallel. It is needed to increase the CPU utilization by overlapping I/O operations and the computation in different programs.

5. Discuss the advantages and disadvantages of user-level threads and kernel-level threads.

User-level threads

Advantages:

- 1. No OS threading-supporting is needed*
- 2. Lightweight, thread switching is inexpensive, no trap to kernel*
- 3. Each process has its own customized scheduling algorithm*

Disadvantages:

- 1. Hard to implement blocking call in threads*
- 2. Hard to deal with page faults*
- 3. No clock interrupt, hard to implement preemption*

Kernel-level threads

Advantages:

- 1. Known to OS scheduler*
- 2. Blocking calls and page faults do not block the whole process*

Disadvantages:

- 1. Slow due to the trap to the kernel*
- 2. Expensive to create and switch*

6. Discuss the goals of CPU scheduling on different computer systems.

Batch systems:

- 1. Maximize throughput*
- 2. Minimize turnaround time*
- 3. Increase CPU utilization*

Interactive systems:

- 1. Minimize response time*
- 2. Meet users' expectations in regard to proportionality*

Real-time systems:

- 1. Meeting deadlines*
- 2. Provide predictable performance*

7. Assume that the following processes are to be executed on a uniprocessor system.

Based on their arrival time and CPU burst, calculate the average turnaround time and response time of these processes under the following scheduling policies:

a. FCFS

b. Round Robin (quantum = 4 and 6)

c. Shortest Job First (preemptive and non-preemptive)

Process	Arrival Time	CPU burst
P1	0	12
P2	0	3
P3	2	7
P4	3	5

Compare the performance of above policies.

a. FCFS-1 (P1 first):

Average turnaround time: $((12-0)+(15-0)+(22-2)+(27-3))/4=17.75$

Average response time: $(0+(12-0)+(15-2)+(22-3))/4=11$

FCFS-2 (P2 first):

Average turnaround time: $((3-0)+(15-0)+(22-2)+(27-3))/4=15.5$

Average response time: $(0+(3-0)+(15-2)+(22-3))/4=8.75$

b. Round Robin (quantum=4)

Time:	0	4	7	11	15	19	22	23
Process:	P1	P2	P3	P4	P1	P3	P4	P1

Average turnaround time: $((27-0)+(7-0)+(22-2)+(23-3))/4=18.5$

Average response time: $((0+(4-0)+(7-2)+(11-3))/4=4.25$

Round Robin (quantum=6)

Time:	0	6	9	15	20	26
Process:	P1	P2	P3	P4	P1	P3

Average turnaround time: $((26-0)+(9-0)+(27-2)+(20-3))/4=19.25$

Average response time: $((0+(6-0)+(9-2)+(15-3))/4=6.25$

c. SJF (preemptive)

Average turnaround time: $((3-0)+(8-3)+(15-2)+(27-0))/4=12$

Average response time: $(0+(3-3)+(8-2)+(15-0))/4=5.25$

SJF (non-preemptive)

Same as the preemptive version.

Comparison:

Best average turnaround time: SJF

Best average response time: RR with quantum=4

Worst average turnaround time: RR with quantum=6

Worst average response time: FCFS (P1 first)

Overall SJF performs best in this data set as it has the best avg. turnaround time and good avg. response time.

8. What are the additional requirements of multiprocessor scheduling compared with uniprocessor scheduling? What are the possible issues?

Two decisions need to be made when scheduling threads on a multiprocessor system: which thread to run and where to run. Multiprocessor scheduling also needs to consider load-balancing between processors. When performs load-balancing, the scheduling needs to take into account the hotness of the cache and the underlying hardware topology, such as the NUMA architecture, hardware hyperthreading.

Different from uniprocessor scheduling, in which threads are dispatched from a single runqueue, multiprocessor scheduling usually uses multiple runqueues, one per processor. It is generally more difficult to balance the load evenly on different processors. Due to the nature of distributed (or self) scheduling, it is usually more difficult to enforce fairness in multiprocessor systems.

9. List different ways to ensure mutual exclusion.

Disabling interrupts, lock variables, strict alternation, Peterson's Solution, TSL, semaphores, Mutexes, and monitors.

10. What are the advantages and disadvantages of busy-waiting and sleep-and-wake approaches for mutual exclusion?

Busy-waiting

Advantages:

- 1. Simple to implement*
- 2. Less overhead if the wait time is short, no context switch*

Disadvantages:

- 1. Possible waste of CPU cycles which otherwise can be used by others*
- 2. Possible priority inversion problems*

Sleep-and-wake

Advantages:

- 1. No wasted CPU cycles*
- 2. No priority inversion problems*

Disadvantages:

- 1. More overhead due to context switches*
- 2. Error-prone*

11. Why semaphore is needed? What are the commonalities and differences between semaphore and mutex.

Semaphores help avoid the lost of wakeup signals. A semaphore is a variable that counts the number of pending wakeups. If properly configured, semaphores allow a predefined number of threads to enter a critical region. Mutex is a simplified version of semaphore (binary semaphore). It is only useful in cases of mutual exclusion. Both

semaphore and mutex provide a sleep-and-wake locking approach to mutual exclusion.

12. In the dining philosophers problem (Page 166, Figure 2-46), explain what the does the function test(i) do (two roles) and how it works.

The function test(i) has two roles:

- 1. Block a philosopher if he/she did not get the forks.*
- 2. Once finishing eating, a philosopher wakes up one of his/her neighbors.*