Homework 2

Due: March 5th, 2015

Submission Instructions

Please type your answers to the following questions and submit the PDF on Aurora/Beowulf. The instructions to submit on Aurora/Beowulf can be found on Moodle. **To submit please use submit hw2 myhw.pdf**

This homework is to be done individually; you can discuss the questions with your classmates, but you should write your answers individually.

Questions in this homework are model questions for the midterm exam. You can expect to get similar questions on the midterm; few of these might repeat as well.

Multiple choice questions (1 point each):

**OS Structures**

1. \_\_\_\_ operating systems are designed primarily to maximize resource utilization.

A) PC

B) Handheld computer

C) Mainframe

D) Network

2. The most common secondary storage device is \_\_\_\_.

A) random access memory

B) solid state disks

C) tape drives

D) magnetic disk

3. A \_\_\_\_ can be used to prevent a user program from never returning control to the operating system.

A) portal

B) program counter

C) firewall

D) timer

4. The two separate modes of operating in a system are \_\_\_\_\_.

A) supervisor mode and system mode

B) kernel mode and privileged mode

C) physical mode and logical mode

D) user mode and kernel mode

￼

5. A message-passing model is \_\_\_\_.

A) easier to implement than a shared memory model for inter-computer communication

B) faster than the shared memory model

C) a network protocol, and does not apply to operating systems

D) only useful for small, simple operating systems

6. Policy \_\_\_\_.

A) determines how to do something

B) determines what will be done

C) is not likely to change across places

D) is not likely to change over time

7. The major difficulty in designing a layered operating system approach is \_\_\_\_.

A) appropriately defining the various layers

B) making sure that each layer hides certain data structures, hardware, and operations from higher-level layers

C) debugging a particular layer

D) making sure each layer is easily converted to modules

8. A microkernel is a kernel \_\_\_\_.

A) containing many components that are optimized to reduce resident memory size

B) that is compressed before loading in order to reduce its resident memory size

C) that is compiled to produce the smallest size possible when stored to disk

D) that is stripped of all nonessential components

9. \_\_\_\_\_ provide(s) an interface to the services provided by an operating system.

A) Shared memory

B) System calls

C) Simulators

D) Communication

10. \_\_\_\_\_ allow operating system services to be loaded dynamically.

A) Virtual machines

B) Modules

C) File systems

D) Graphical user interfaces

**Processes:**

11. The \_\_\_\_ of a process contains temporary data such as function parameters, return addresses, and local variables.

A) text section

B) data section

C) program counter

D) stack

12. The \_\_\_\_\_\_\_\_ refers to the number of processes in memory.

A) process count

B) long-term scheduler

C) degree of multiprogramming

D) CPU scheduler

13. Which of the following statements is true?\_\_\_\_\_\_

A) Shared memory is typically faster than message passing.

B) Message passing is typically faster than shared memory.

C) Message passing is most useful for exchanging large amounts of data.

D) Shared memory is far more common in operating systems than message passing.

14. A process control block \_\_\_\_.

A) includes information on the process's state

B) stores the address of the next instruction to be processed by a different process

C) determines which process is to be executed next

D) is an example of a process queue

15. A \_\_\_\_\_\_\_\_\_ saves the state of the currently running process and restores the state of the next process to run.

A) save-and-restore

B) state switch

C) context switch

D) none of the above

**Threads:**

16. The \_\_\_\_ multithreading model multiplexes map many user-level threads to a smaller or equal number of kernel threads.

A) many-to-one

B) one-to-one

C) many-to-many

D) many-to-some

17. Thread-local storage is data that \_\_\_\_.

A) is not associated with any process

B) has been modified by the thread, but not yet updated to the parent process

C) is generated by the thread independent of the thread's process

D) is unique to each thread

18. LWP is \_\_\_\_.

A) short for lightweight processor

B) placed between system and kernel threads

C) placed between user and kernel threads

D) common in systems implementing one-to-one multithreading models

19. Most modern operating systems provide by default \_\_\_\_.

A) one-to-one model

B) many-to-one model

C) one-to many-model

D) many-to-many model

20. A \_\_\_\_ provides an API for creating and managing threads.

A) set of system calls

B) multicore system

C) thread library

D) multithreading model

**CPU Scheduling:**

21. \_\_\_\_ is the number of processes that are completed per time unit.

A) CPU utilization

B) Response time

C) Turnaround time

D) Throughput

22. \_\_\_\_ scheduling is approximated by predicting the next CPU burst with an exponential average of the measured lengths of previous CPU bursts.

A) Multilevel queue

B) RR

C) FCFS

D) SJF

23. The \_\_\_\_ scheduling algorithm is designed especially for time-sharing systems.

A) SJF

B) FCFS

C) RR

D) Multilevel queue

24. Which of the following scheduling algorithms must be non-preemptive? \_\_\_\_

A) SJF

B) RR

C) FCFS

D) priority algorithms

**Synchronization:**

25. An instruction that executes atomically \_\_\_\_.

A) must consist of only one machine instruction

B) executes as a single, uninterruptible unit

C) cannot be used to solve the critical section problem

D) all of the above

26. A(n) \_\_\_\_\_\_\_ refers to where a process is accessing/updating shared data.

A) critical section

B) entry section

C) mutex

D) test-and-set

27. \_\_\_\_\_ can be used to prevent busy waiting when implementing a lock.

A) Spinlocks

B) Waiting queues

C) Mutex lock

D) Allowing the wait() operation to succeed

Descriptive Questions:

**Processes:**

28. (a) What is a process? (3)

(b) What are the five states that a process can go through in its lifetime? (5)

(c) Draw the process state transition diagram, as discussed in the class. (5)

(d) Can a process transition from waiting state to running state? Why or why not? (5)

29. Including the initial parent process, how many processes are created by the program shown below? Draw the corresponding process tree hierarchy (10)

#include <stdio.h>

#include <unistd.h>

int main(){

/\*fork a child process \*/

fork();

/\*fork a child process \*/

fork();

/\*fork a child process \*/

fork();

return 0;

}

**Threads**

30. Which of the following components of program state are shared across threads in a

multithreaded process? (2 + 2 + 2 + 2 + 2)

(a) Register values

(b) Heap memory

(c) Global variables

(d) Stack memory

(e) Program code

31. Can a multithreaded program using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system? Why or why not? (10)

**CPU Scheduling:**

32. Following is the arrival and processing times of processes in a system:

<Process A, arrival time = 0, processing time = 3>

<Process B, arrival time = 1, processing time = 6>

<Process C, arrival time = 4, processing time = 4>

<Process D, arrival time = 6, processing time = 2>

For the above system, what is the wait time for each process using: (2.5 + 2.5 + 2.5 + 2.5)

(a) First-Come First-Served

(b) Shortest Job First

(c) Shortest Remaining Time First

(d) Round Robin (time quantum = 2)

**Synchronization:**

33. Consider a system where several threads are competing for accessing the critical section.

Each of this thread executes the following code for synchronization:

do {

while (TestAndSet(&lock))

; // do nothing

//Critical Section Code

lock = FALSE;

//remainder section

} while (TRUE);

Here “TestAndSet()” is the atomic read-modify-write instruction provided by the system architecture; we talked about this instruction in class. Answer the following questions based on above:

* (i) Does the above synchronization code ensure mutual exclusion for access to the critical section? Why or why not? (2 + 5)
* (ii) A synchronization solution is said to provide bounded waiting if there exists a bound, or limit, on the number of times that other processes/threads are allowed to enter their critical sections after any given process/thread has made a request to enter its critical section and before that request is granted. Does the above synchronization solution guarantee bounded waiting? Why or why not? (3 + 5)