## Homework assignment 4 (based on chapters 4 and 5)

Chapter 4 Q&A

1. Which of the following items are shared across multiple threads belonging to the same process?
A) code, data, files
B) registers
C) stack
D) all of the above

Ans: A

2. In a multithreaded server architecture, which of the following is used to service a new user request?
A) a new created thread
B) a new created process
C) the same process for prior users
D) none of the above

Ans: A

3. Which of the following benefits go to multithreaded programming?
A) responsiveness
B) resource sharing
C) economy
D) scalability
E) all of the above

Ans: E

4. Which of the following refers to the capability to allow multiple tasks make progress on a single processor system?
A) concurrency
B) parallelism
C) data parallelism
D) task parallelism

Ans: A

5. _____ is a formula that identifies potential performance gains from adding additional computing cores to an application that has a parallel and serial component.

A) Task parallelism
B) Data parallelism
C) Data splitting
D) Amdahl's Law

Ans: D

6. _____ is not considered a challenge when designing applications for multicore systems.
A) Deciding which activities can be run in parallel
B) Ensuring there is a sufficient number of cores
C) Determining if data can be separated so that it is accessed on separate cores
D) Identifying data dependencies between tasks.

Ans: B

7. Which of the following models are possible for the relationship between the user threads and kernel threads?
A) many-to-one model
B) one-to-one model
C) many-to-many model
D) two-level model
E) all of the above

Ans: E

8. ____ is a thread library for Solaris that maps many user-level threads to one kernel thread.
A) Pthreads
B) Green threads
C) Sthreads
D) Java threads

Ans: B

9. The _____ model maps each user-level thread to one kernel thread.
A) many-to-many
B) two-level
C) one-to-one
D) many-to-one

Ans: C

10. The _____ model multiplexes many user-level threads to a smaller or equal number of kernel threads.
A) many-to-many
B) two-level
C) one-to-one
D) many-to-one

Ans: A

11. Which of the following is a function that can be provided by Pthreads API for constructing a multithreaded program?
A) pthread attr init
B) pthread_create
C) pthread_join
D) all of the above

Ans: D

12. In Pthreads, a parent uses the pthread_join() function to wait for its child thread to complete. What is the equivalent function in WinAPI?
A) win32_join()
B) wait()
C) WaitForSingleObject()

D) join()

Ans: C

13. Which of the following statements regarding threads is false?
A) Sharing is automatically provided in Java threads.
B) Both Pthreads and WinAPI threads share global data.
C) The start() method actually creates a thread in the Java virtual machine.
D) The Java method join() provides similar functionality as the WaitForSingleObject in Win32.

Ans: A

14. The most common technique for writing multithreaded Java programs is _____.
A) extending the Thread class and overriding the run() method
B) implementing the Runnable interface and defining its run() method
C) designing your own Thread class
D) using the CreateThread() function

Ans: B

15. Which of the following is a method for implicit threading?
A) thread pools
B) OpenMP
C) all of the above

Ans: C

16. Which of the following options exist to deliver signals in multithreaded program?
A) deliver the signal to the thread to which the signal applies
B) deliver the signal to every thread in the process
C) deliver the signal to certain threads in the process
D) assign a specific thread to receive all signals for the process
E) all of the above

Ans: E

17. Which of the following options to deliver signals in multithreaded program should be applied to an asynchronous signal that terminates a process (<control><C>, for example)?
A) deliver the signal to the thread to which the signal applies
B) deliver the signal to every thread in the process

C) deliver the signal to certain threads in the process
D) assign a specific thread to receive all signals for the process
E) all of the above

Ans: B

18. Which of the following options to deliver signals in multithreaded program should be applied to a synchronous signal?
A) deliver the signal to the thread to which the signal applies
B) deliver the signal to every thread in the process
C) deliver the signal to certain threads in the process
D) assign a specific thread to receive all signals for the process
E) all of the above

Ans: A

19. To associate each thread created using an implicit technique such as a thread pool, with its unique transaction identifier, we could use ____?
A) global variable
B) local variable
C) static data
D) thread-local storage

Ans: D

20. 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

Ans: C

21. Which are included in the context of a thread?
A) register set
B) stacks
C) private storage area
D) all of the above

Ans: D

22. Which of the following information is shared when the flag CLONE_VM in Linux clone() system call is set up?
A) file-system information
B) memory space
C) signal handlers
D) set of open files

**Essay Questions**

1. Why should a web server not run as a single-threaded process?

Ans: For a web server that runs as a single-threaded process, only one client can be serviced at a time. This could result in potentially enormous wait times for a busy server.

2. List the four major categories of the benefits of multithreaded programming. Briefly explain each.

Ans: The benefits of multithreaded programming fall into the categories: responsiveness, resource sharing, economy, and utilization of multiprocessor architectures. Responsiveness means that a multithreaded program can allow a program to run even if part of it is blocked. Resource sharing occurs when an application has several different threads of activity within the same address space. Threads share the resources of the process to which they belong. As a result, it is more economical to create new threads than new processes. Finally, a single-threaded process can only execute on one processor regardless of the number of processors actually

present. Multiple threads can run on multiple processors, thereby increasing efficiency.
3. Distinguish between parallelism and concurrency.

Ans: A parallel system can perform more than one task simultaneously. A concurrent system supports more than one task by allowing multiple tasks to make progress.

4. Multicore systems present certain challenges for multithreaded programming. Briefly describe these challenges.

Ans: Multicore systems have placed more pressure on system programmers as well as application developers to make efficient use of the multiple computing cores. These challenges include determining how to divide applications into separate tasks that can run in parallel on the different cores. These tasks must be balanced such that each task is doing an equal amount of work. Just as tasks must be separated, data must also be divided so that it can be accessed by the tasks running on separate cores. So that data can safely be accessed, data dependencies must be identified and where such dependencies exist, data accesses must be synchronized to ensure the safety of the data. Once all such challenges have been met, there remains considerable challenges testing and debugging such applications.

5. Distinguish between data and task parallelism.

Ans: Data parallelism involves distributing subsets of the same data across multiple computing cores and performing the same operation on each core. Task parallelism involves distributing tasks across the different computing cores where each task is performing a unique operation.

6. What are the two different ways in which a thread library could be implemented?

Ans: The first technique of implementing the library involves ensuring that all code and data structures for the library reside in user space with no kernel support. The other approach is to implement a kernel-level library supported directly by the operating system so that the code and data structures exist in kernel space.

Chapter 5 Q&A

**Multiple Choice Questions**

1. The ready queue can be implemented as a _____.
A) FIFO queue
B) priority queue
C) tree

D) unordered linked list
E) all of the above

Ans: E

2. Which of the following circumstances can cooperative scheduling take place?
A) when a process switches from the running state to the waiting state
B) when a process switches from the running state to the ready state
C) when a process switches from the waiting state to the ready state
D) none of the above

Ans: A

3. Which of the following circumstances can preemptive scheduling take place?
A) when a process switches from the running state to the waiting state
B) when a process switches from the waiting state to the ready state
C) when a process terminates
D) none of the above

Ans: B

4. Which of the following items does not belong to the function of a dispatcher?
A) switching context from one process to another
B) selecting a process among the available ones in the ready queue
C) switching to user mode
D) jumping to the proper location in the user program to resume that program

Ans: B

5. Assume process $P_0$ and $P_1$ are the process before and after a context switch, and $PCB_0$ and $PCB_1$ are respectively their process control block. Which of the following time units are included inside the dispatch latency?
A) $P_0$ executing
B) save state into $PCB_0$, and restore state from $PCB_1$
C) $P_1$ executing
D) all of the above

Ans: B

6. Which of the following criteria is more important for an interactive system?
A) CPU utilization
B) Response time
C) Turnaround time
D) Throughput

Ans: B

7. Which of the following criteria is more important from the point of view of a particular process?
A) CPU utilization
B) Response time
C) Turnaround time
D) Throughput

Ans: C

8. For interactive systems, it is more important to minimize _____.
A) the average response time
B) the average turnaround time
C) the variance in response time
D) the waiting time

Ans: C

9. Which of the following scheduling algorithm may suffer from convoy effect?
A) SJF
B) FCFS
C) RR
D) Multilevel queue

Ans: B

10. Which of the following scheduling algorithms must be nonpreemptive?
A) SJF
B) RR
C) FCFS
D) priority algorithms

Ans: C

11. I/O-bound program typically has many short _____ and a CPU-bound program might have a few long _____.
A) I/O burst, CPU burst
B) I/O burst, I/O burst
C) CPU burst, CPU burst
D) CPU burst, I/O burst

Ans: C

12. The _____ occurs in first-come-first-served scheduling when a process with a long CPU burst occupies the CPU.
A) dispatch latency
B) waiting time
C) convoy effect
D) system-contention scope

Ans: C

13. Which of the following scheduling algorithms gives the minimum average waiting time for a given set of processes?
A) SJF
B) FCFS
C) RR
D) Multilevel queue

Ans: A

14. Shortest-remaining-time-first scheduling is the preemptive version of _____?
A) SJF
B) FCFS
C) RR
D) Multilevel queue

Ans: A

15. Which of the following scheduling algorithms gives the minimum average response time?
A) SJF
B) FCFS
C) RR
D) Multilevel queue

Ans: C

16. A significant problem with priority scheduling algorithms is _____.
A) complexity
B) starvation
C) determining the length of the next CPU burst
D) determining the length of the time quantum

Ans: B

17. If the time quantum get too large, RR scheduling degenerates to _____?
A) SJF
B) FCFS
C) Shortest-remaining-time-first
D) Multilevel queue

Ans: B

18. Which of the following can be a solution to the problem of indefinite blockage of low-priority processes?
A) Aging
B) Starvation
C) Multilevel queue
D) All of the above

Ans: A

19. Which of the following processes usually have highest priority?
A) real-time processes
B) system processes
C) interactive processes
D) batch processes

Ans: A

20. Which of the following is allowed on Linux and Mac OS X systems?
A) only PTHREAD_SCOPE_PROCESS
B) only PTHREAD_SCOPE_SYSTEM
C) Both PTHREAD_SCOPE_PROCESS and PTHREAD_SCOPE_SYSTEM
D) none of the above

Ans: B

21. Which of the following system architectures involves multiprocessor scheduling?
A) multicore CPUs
B) multithreaded cores
C) NUMA systems
D) heterogeneous multiprocessing
E) all of the above

Ans: E

22. Which of the following is preemptive?
A) rate-monotonic scheduling
B) earliest-deadline-first scheduling
C) both of the above
D) none of the above

Ans: C

23. Which of the following POSIX API sets the scheduling policy?
A) pthread_attr_getsched_policy
B) pthread_attr_setsched_policy
C) pthread_attr_getscope
D) pthread_attr_setscope

Ans: B

**Essay Questions**

1. Explain the concept of a CPU–I/O burst cycle.

Ans: The lifecycle of a process can be considered to consist of a number of bursts belonging to two different states. All processes consist of CPU cycles and I/O operations. Therefore, a process can be modeled as switching between bursts of CPU execution and I/O wait.

2. What role does the dispatcher play in CPU scheduling?

Ans: The dispatcher gives control of the CPU to the process selected by the short-term scheduler. To perform this task, a context switch, a switch to user mode, and a jump to the proper location in the user program are all required. The dispatch should be made as fast as possible. The time lost to the dispatcher is termed dispatch latency.

3. Explain the difference between response time and turnaround time. These times are both used to measure the effectiveness of scheduling schemes.

Ans: Turnaround time is the sum of the periods that a process is spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O. Turnaround time essentially measures the amount of time it takes to execute a process. Response time, on the other hand, is a measure of the time that elapses between a request and the first response produced.

4. Explain the process of starvation and how aging can be used to prevent it.

Ans: Starvation occurs when a process is ready to run but is stuck waiting indefinitely for the CPU. This can be caused, for example, when higher-priority processes prevent low-priority processes from ever getting the CPU. Aging involves gradually increasing the priority of a process so that a process will eventually achieve a high enough priority to execute if it waited for

a long enough period of time.

5. What effect does the size of the time quantum have on the performance of an RR algorithm?

Ans: At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy. If the time quantum is extremely small, the RR approach is called processor sharing and creates the appearance that each of $n$ processes has its own processor running at $1/n$ the speed of the real processor.

6. Explain the fundamental difference between asymmetric and symmetric multiprocessing.

Ans: In asymmetric multiprocessing, all scheduling decisions, I/O, and other system activities are handled by a single processor, whereas in SMP, each processor is self-scheduling.

7. Describe two general approaches to load balancing.

Ans: With push migration, a specific task periodically checks the load on each processor and — if it finds an imbalance—evenly distributes the load by moving processes from overloaded to idle or less-busy processors. Pull migration occurs when an idle processor pulls a waiting task from a busy processor. Push and pull migration are often implemented in parallel on load-balancing systems.