WAYGROUND formerly Quizizz **Worksheets**

Name

OS UNIT-2 PVPSIT

Total questions: 47

Worksheet time: 4hrs 55mins

Instructor name: Mr. PRASHANT ATMAKURI

Class

Date

1.

3.1 Using the program shown in Figure 3.30, explain what the output will be at LINE A.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
pid_t pid;

  pid = fork();

  if (pid == 0) { /* child process */
    value += 15;
    return 0;
  }
  else if (pid > 0) { /* parent process */
    wait(NULL);
    printf("PARENT: value = %d",value); /* LINE A */
    return 0;
  }
}
```

Figure 3.30 What output will be at Line A?

109

Using the program shown in Figure 3.30, what is printed at LINE A by the parent process after it calls wait(NULL)? #include #include #include int value = 5; int main() { pid_t pid; pid = fork(); if (pid == 0) { /* child process */ value += 15; return 0; } else if (pid > 0) { /* parent process */ wait(NULL); printf("PARENT: value = %d", value); /* LINE A */ return 0; } }

a) PARENT: value = 5

b) PARENT: value = 15

c) No output is printed

d) PARENT: value = 20

2. Including the initial parent process, how many processes are created by the program shown in Figure 3.31?

a) 9

b) 7

c) 6

d) 8

3. Original versions of Apple's mobile iOS operating system provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

   a) Concurrent processes may need to communicate, requiring the operating system to provide interprocess communication methods

   b) Concurrency guarantees higher battery life because processes share CPU efficiently

   c) Limited memory on mobile devices means poorly managed processes can negatively impact overall performance, so the OS must support concurrent processes with careful memory management

   d) Concurrent processing eliminates the need for context switching, simplifying OS design

   e) The CPU scheduler must be aware of different concurrent processes and choose an appropriate scheduling algorithm

4. Some computer systems provide multiple register sets. Describe what happens when a context switch occurs if the new context is already loaded into one of the register sets. What happens if the new context is in memory rather than in a register set and all the register sets are in use?

   a) If the new context is in memory and all register sets are in use, a context in a register set must be chosen as a victim, moved to memory, and the new context loaded into the freed set

   b) If the new context is already loaded into a register set, the CPU current-register-set pointer is changed to point to that set, taking very little time

   c) If the new context is already loaded into a register set, it must still be reloaded from memory, which takes considerable time

   d) If the new context is in memory and all register sets are in use, the CPU can execute both contexts simultaneously without moving registers

5. When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

   a) Shared memory segments

   b) Stack

   c) Heap

   d) Open files

6. Consider the “exactly once” semantic with respect to the RPC mechanism. Does the algorithm for implementing this semantic execute correctly

   a) No, RPC always executes a procedure multiple times regardless of the algorithm
   b) Yes, the algorithm guarantees exactly-once execution under all conditions
   c) Yes, but only when there are no network delays and the server never fails
   d) No, it cannot be guaranteed in the presence of network or server failures, so the algorithm does not execute correctly in general

7. In a remote procedure call (RPC) system, consider the case where the acknowledgment (ACK) sent back to the client is lost due to a network problem. Describe the sequence of client and server actions that preserves "exactly once" semantics, and identify the correct behavior.

   a) The client never retries the RPC; the server eventually times out and discards the request.
   b) The client resends the RPC with a later timestamp after timeout; the server uses timestamps to detect the duplicate, does not re-execute the procedure, and sends a second ACK.
   c) The client resends the RPC with the same timestamp; the server executes the procedure again to ensure delivery.
   d) The server automatically re-executes the RPC on receiving a duplicate to ensure completion, and sends no ACK.

8. Assume a distributed system is susceptible to server failure. Which mechanism is required on the server to guarantee "exactly once" semantics for the execution of RPCs after a crash?

   a) Use client-side caching of requests without any server-side tracking of execution.
   b) Maintain a stable-storage log of received RPCs, whether they were performed, and their results, and consult it on recovery.
   c) Rely on periodic checkpointing of memory only, without recording individual RPCs or outcomes.
   d) Ignore RPCs received during recovery and require all clients to resend operations.

9.

```
#include <stdio.h>
#include <unistd.h>

int main()
{
   /* fork a child process */
   fork();

   /* fork another child process */
   fork();

   /* and fork another */
   fork();

   return 0;
}
```

Figure 3.31 How many processes are created?

How many processes are created?

a) 6
b) 4
c) 7
d) 8

10. Select all scenarios in which multithreading provides better performance than a single-threaded solution.

a) An interactive GUI program such as a debugger where one thread monitors user input, another represents the running application, and a third thread monitors performance
b) A simple script that prints a single line of text and exits
c) A web server that services each request in a separate thread
d) A parallelized application such as matrix multiplication where various parts of the matrix are worked on in parallel
e) A single CPU-bound loop with no I/O and no independent subtasks

11. Using Amdahl's Law, calculate the speedup of an application with a 60 percent parallel component on two processing cores.

a) 1.42
b) 1.82
c) 1.60
d) 1.20

12. Using Amdahl's Law, calculate the speedup of an application with a 60 percent parallel component on four processing cores.

a) 1.82
b) 1.72
c) 1.42
d) 1.60

13. For the multithreaded web server described in Section 4.1, identify the form of parallelism it exhibits.

a) Pipeline parallelism
b) Speculative parallelism
c) Data parallelism
d) Task parallelism

14. Which statement correctly distinguishes user-level threads from kernel-level threads?

a) Both user-level and kernel-level threads are unknown by the kernel
b) Kernel-level threads are unknown by the kernel, whereas the kernel is aware of user-level threads
c) User-level threads and kernel-level threads are identical in how the kernel manages them
d) User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads

15. Describe the actions taken by a kernel to context-switch between kernel-level threads.

a) Save the CPU registers of the thread being switched out and restore the CPU registers of the new thread being scheduled
b) Flush the TLB and rebuild page tables without touching registers
c) Swap user-level stacks and update thread library metadata only
d) Write the old thread’s state to disk and reload the new thread’s code segment

16. What resources are used when a thread is created, and how do they differ from those used when a process is created?

a) Thread creation requires allocating a full process control block identical to that of a process, and process creation reuses the thread’s stack
b) Thread creation primarily duplicates the process’s memory map, while process creation only adds a lightweight thread descriptor
c) Thread creation allocates a small data structure (register set, stack, priority), whereas process creation requires a larger process control block plus a memory map, open-file list, and environment variables
d) Thread creation consumes more memory than process creation because each thread needs its own address space and open-file table

17. Assume an operating system maps user-level threads to the kernel using a many-to-many model through LWPs and allows developers to create real-time threads. Is it necessary to bind a real-time thread to an LWP?

a) No; real-time threads are always immediately scheduled regardless of LWP attachment

b) Yes; binding prevents the kernel from preempting lower-priority threads

c) Yes; binding ensures the real-time thread can run with minimal delay instead of waiting to be attached to an LWP

d) No; binding increases latency because the LWP must be reassigned on each I/O operation

18. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes.

Given n processes to be scheduled on one processor, how many different schedules are possible?

Give a formula in terms of n .

a) n

b) n^2

c) 2^n

d) $n!$

19. Explain the difference between preemptive and nonpreemptive scheduling.

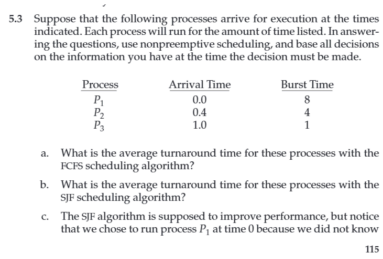
a) In preemptive scheduling, a process may be interrupted during its CPU burst and the CPU reallocated to another process.

b) In nonpreemptive scheduling, a process continues its current CPU burst and relinquishes the CPU only after finishing that burst.

c) Nonpreemptive scheduling allows the scheduler to take the CPU away from a process at any moment.

d) Preemptive scheduling requires all processes to arrive at time 0 to be considered.

20.



5.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know

115

Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made. What is the average turnaround time for these processes with the FCFS scheduling algorithm? Process data: P1 arrival time 0.0 and burst time 8; P2 arrival time 0.4 and burst time 4; P3 arrival time 1.0 and burst time 1.

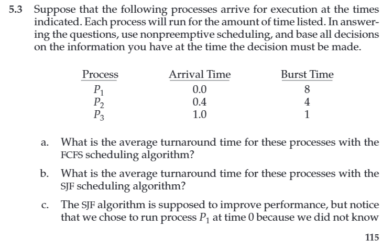
a) 9.00 time units

b) 11.00 time units

c) 10.53 time units

d) 8.00 time units

21.



5.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know

115

Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made. What is the average turnaround time for these processes with the SJF scheduling algorithm? Process data: P1 arrival time 0.0 and burst time 8; P2 arrival time 0.4 and burst time 4; P3 arrival time 1.0 and burst time 1.

a) 9.53 time units

b) 12.60 time units

c) 10.00 time units

d) 8.67 time units

22. Consider the following set of processes, with the length of the CPU burst time given in milliseconds. All processes arrive at time 0 in the order P1, P2, P3, P4, P5. Process table: P1 burst 2, priority 2; P2 burst 1, priority 1; P3 burst 8, priority 4; P4 burst 4, priority 2; P5 burst 5, priority 3. Using FCFS scheduling, what is the execution order of the processes?

a) P1, P2, P3, P4, P5

b) P3, P5, P1, P4, P2

c) P2, P1, P4, P5, P3

d) P1, P3, P2, P4, P5

23. Consider the same process set (all arrive at time 0). Using non-preemptive SJF scheduling, what is the execution order of the processes?

a) P1, P2, P3, P4, P5
b) P3, P5, P1, P4, P2
c) P2, P1, P4, P5, P3
d) P2, P4, P1, P5, P3

24. For the same process set, use non-preemptive priority scheduling where a larger priority number implies a higher priority. What is the execution order of the processes?

a) P3, P5, P1, P4, P2
b) P2, P1, P4, P5, P3
c) P5, P3, P4, P1, P2
d) P1, P2, P3, P4, P5

25. Using Round Robin scheduling with time quantum 2 milliseconds and the same process set (all arrive at time 0), which process completes third?

a) P5
b) P3
c) P4
d) P2

26. Using FCFS on the given process set (all arrive at time 0), what is the turnaround time of process P4? Turnaround time is finishing time minus arrival time.

a) 7 ms
b) 15 ms
c) 13 ms
d) 19 ms

27. Using non-preemptive SJF on the given process set, what is the turnaround time of process P5? Turnaround time is finishing time minus arrival time.

a) 7 ms
b) 20 ms
c) 12 ms
d) 15 ms

28. Using Round Robin with quantum 2 ms on the given process set, what is the turnaround time of process P3? Turnaround time is finishing time minus arrival time.

a) 20 ms

b) 18 ms

c) 15 ms

d) 11 ms

29. Across the four scheduling algorithms specified (FCFS, SJF, non-preemptive priority with larger number = higher priority, and RR with quantum 2 ms), which results in the minimum average waiting time over all processes?

a) Non-preemptive priority

b) SJF

c) Round Robin (quantum 2 ms)

d) FCFS

30. Using Round Robin with quantum 2 ms on the given process set, what is the waiting time of process P5? Waiting time is turnaround time minus burst time.

a) 9 ms

b) 13 ms

c) 15 ms

d) 12 ms

31. Compute the average turnaround time if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. Turnaround time is finishing time minus arrival time.

a) 9.53

b) 6.86

c) 10.53

d) 11.00

32. According to the turnaround time table, what is the turnaround time of P1 under the Priority scheduling algorithm?

a) 2
b) 3
c) 20
d) 15

33. According to the turnaround time table, what is the turnaround time of P5 under FCFS?

a) 20
b) 18
c) 12
d) 15

34. According to the waiting time table (turnaround time minus burst time), what is the waiting time of P4 under Round Robin (RR)?

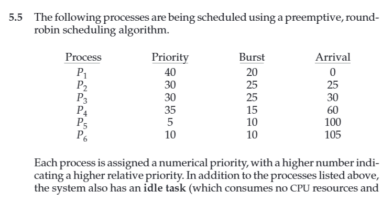
a) 3
b) 11
c) 15
d) 9

35. From the waiting time table, what is the waiting time of P3 under the Priority scheduling algorithm?

a) 12
b) 3
c) 0
d) 8

36. Based on the provided statement and waiting time table, which scheduling algorithm has the shortest wait time overall?

a) Priority
b) SJF
c) Round Robin (RR)
d) FCFS

37. 

5.5 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm.

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed above, the system also has an **idle task** (which consumes no CPU resources and

The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. A higher numerical priority indicates a higher relative priority. Which process has the highest priority? Options show process identifiers only.

a) P1
b) P2
c) P6
d) P4

38.

5.5 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm.

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed above, the system also has an **idle task** (which consumes no CPU resources and

Using the preemptive round-robin process table, which process has priority 30 and arrival time 25 ?

a) P2

b) P4

c) P3

d) P1

39. In the same schedule, which set of turnaround times for the processes is correct?

a) P1: 20; P2: 50; P3: 50; P4: 10; P5: 25; P6: 10

b) P1: 25; P2: 60; P3: 45; P4: 20; P5: 30; P6: 15

c) P1: 20; P2: 55; P3: 60; P4: 15; P5: 20; P6: 10

d) P1: 20; P2: 40; P3: 35; P4: 0; P5: 10; P6: 0

40. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

a) It allows frequently serviced interactive processes to run in a small time quantum while processes needing less frequent servicing run in larger quanta, reducing context switches and improving overall efficiency.

b) It eliminates the need for multiple queues by merging all processes into one queue with a fixed quantum.

c) It prevents preemption entirely, ensuring long CPU bursts complete without interruption.

d) It guarantees identical response time for all processes by using the same quantum everywhere, which simplifies scheduling.

41. What relation holds between Priority scheduling and Shortest Job First (SJF) when considering sets of parameterized algorithms?

a) SJF is a special case within the Priority scheduling set when priority is based on predicted next CPU burst length.

b) Priority scheduling is a special case of SJF with infinite time quantum.

c) They are equivalent algorithms under all parameter choices.

d) No general relation exists between the sets.

42. What relation holds between Multilevel feedback queues (MLFQ) and First-Come, First-Served (FCFS) when considering sets of parameterized algorithms?

a) They are equivalent under all parameterizations.
b) No general relation exists between the sets.
c) FCFS is a special case within the MLFQ set (for example, with a single queue and an infinite time quantum).
d) MLFQ is a special case of FCFS when time quantum is set to zero.

43. What relation holds between Priority scheduling and First-Come, First-Served (FCFS) when considering sets of parameterized algorithms?

a) They are equivalent under all parameterizations.
b) No general relation exists between the sets.
c) Priority scheduling is a special case of FCFS with an infinite time quantum.
d) FCFS can be seen as a special case within Priority scheduling when all processes share equal priority and are served in arrival order.

44. What relation holds between Round Robin (RR) and Shortest Job First (SJF) when considering sets of parameterized algorithms?

a) SJF is a special case of RR with an infinite time quantum.
b) They are equivalent under all parameterizations.
c) No general relation holds; RR with an infinite time quantum reduces to FCFS, not SJF.
d) RR is a special case of SJF when the quantum equals the shortest burst.

45. A CPU scheduling algorithm favors processes that have used the least processor time in the recent past. Explain why this policy favors I/O-bound programs and why it does not permanently starve CPU-bound programs.

a) It favors I/O-bound programs because their CPU bursts are short, keeping recent CPU time low; CPU-bound programs do not starve because I/O-bound programs frequently yield the CPU to perform I/O.
b) It favors I/O-bound programs and permanently starves CPU-bound programs because they always use more CPU time.
c) It treats I/O-bound and CPU-bound programs equally, and starvation can occur randomly for either type.
d) It favors CPU-bound programs because they accumulate more processor time; I/O-bound programs starve due to frequent I/O waits.

46. Distinguish between PCS and SCS scheduling.

a) PCS scheduling handles kernel processes only, whereas SCS scheduling handles user processes only.

b) PCS scheduling is performed by the operating system for kernel threads, whereas SCS scheduling is done by a user-level thread library.

c) PCS and SCS scheduling are identical on all systems regardless of the one-to-one or many-to-one model.

d) PCS scheduling is local to a process (user-level thread library maps threads onto available LWPs), whereas SCS scheduling occurs in the operating system to schedule kernel threads.

47. The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priority (higher numbers mean lower priority) and recalculates priorities once per second using $\text{Priority} = \frac{\text{recent CPU usage}}{2} + \text{base}$, where base $= 60$. Assume recent CPU usage for processes P_1 is 40, for P_2 is 18, and for P_3 is 10. Which option correctly gives the new priorities for P_1, P_2, and P_3, and states how this scheduler changes the relative priority of CPU-bound processes?

a) $(70, 69, 65)$ and it lowers the relative priority of CPU-bound processes.

b) $(80, 60, 65)$ and it raises the relative priority of CPU-bound processes.

c) $(80, 69, 65)$ and it raises the relative priority of CPU-bound processes.

d) $(80, 69, 65)$ and it lowers the relative priority of CPU-bound processes.

Answer Keys

1. a) PARENT: value = 5
2. d) 8
3. e) The CPU scheduler must be aware of different concurrent processes and choose an appropriate scheduling algorithm , a) Concurrent processes may need to communicate, requiring the operating system to provide interprocess communication methods , c) Li m m de m po m pr ca ne im ov pe sc m co pr wi m m
4. b) If the new context is already loaded into a register set, the CPU current-register-set pointer is changed to point to that set, taking , a) If the new context is in memory and all register sets are in use, a context in a register set must be chosen as a victim, moved to memory, and the new context loaded into the freed set
5. a) Shared memory segments
6. d) No, it cannot be guaranteed in the presence of network or server failures, so the algorithm does not execute correctly in general

very little time

7. b) The client resends the RPC with a later timestamp after timeout; the server uses timestamps to detect the duplicate, does not re-execute the procedure, and sends a second ACK.

8. b) Maintain a stable-storage log of received RPCs, whether they were performed, and their results, and consult it on recovery.

9. d) 8

10. c) A web server that services each request in a separate thread , d) A parallelized application such as matrix multiplication where various parts of the matrix are worked on in parallel

11. a) 1.42 a) interactive GUI program such as a debugger where one thread monitors user input, another represents the running application, and a third thread monitors performance

12. a) 1.82

13. c) Data parallelism

14. d) User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads

15. a) Save the CPU registers of the thread being switched out and restore the CPU registers of the new thread being scheduled

16. c) Thread creation allocates a small data structure (register set, stack, priority), whereas process creation

17. c) Yes; binding ensures the real-time thread can run with minimal delay instead of waiting to be attached to an LWP

18. d) $n!$

requires a larger process control block plus a memory map, open-file list, and environment variables

19. a) In preemptive scheduling, a process may be interrupted during its CPU burst and the CPU reallocated to another process., b) In nonpreemptive scheduling, a process continues its current CPU burst and relinquishes the CPU only after finishing that burst.

20. c) 10.53 time units

21. a) 9.53 time units

22. a) P1, P2, P3, P4, P5

23. c) P2, P1, P4, P5, P3

24. a) P3, P5, P1, P4, P2

25. c) P4

26. b) 15 ms

27. c) 12 ms

28. a) 20 ms

29. b) SJF

30. b) 13 ms

31. a) 9.53

32. d) 15

33. a) 20

34. d) 9

35. c) 0

36. b) SJF

37. a) P1

38. a) P2

39. c) P1: 20; P2: 55; P3: 60; P4: 15; P5: 20; P6: 10

40. a) It allows frequently serviced interactive processes to run in a small time quantum while processes needing less frequent servicing run in larger quanta, reducing context switches and

41. a) SJF is a special case within the Priority scheduling set when priority is based on predicted next CPU burst length.

42. c) FCFS is a special case within the MLFQ set (for example, with a single queue and an infinite time quantum).

improving overall efficiency.

43. d) FCFS can be seen as a special case within Priority scheduling when all processes share equal priority and are served in arrival order.

44. c) No general relation holds; RR with an infinite time quantum reduces to FCFS, not SJF.

45. a) It favors I/O-bound programs because their CPU bursts are short, keeping recent CPU time low; CPU-bound programs do not starve because I/O-bound programs frequently yield the CPU to perform I/O.

46. d) PCS scheduling is local to a process (user-level thread library maps threads onto available LWPs), whereas SCS scheduling occurs in the operating system to schedule kernel threads.

47. d) $(80, 69, 65)$ and it lowers the relative priority of CPU-bound processes.