CS 202: Advanced Operating Systems, Spring 2022

89

Midterm Exam

Name & UCR Login ID: [redacted]

I. True and False (2 points each)

28

1. ([✓ T] / F) OS has exclusive access to hardware. T

2. (T / [✓ F]) Processes in the user space can execute halt instructions directly. F

3. (T / [✓ F]) Some interrupt handlers can be implemented in the user space. F

4. ([✓ T] / F) To issue a system call in xv6, a user-level process passes the function pointer of the corresponding kernel function to the kernel.

5. (T / [✓ F]) xv6 is considered a library OS. F

6. (T / [✓ F]) xv6 implements priority-based scheduling by default. F

-2 ✗ 7. (T / [✓ F]) Only one thread or process can be in the RUNNING state on a *single* CPU. T

✗ 8. ([✓ T] / F) Context switching between the threads of the same process does **not** need to save and restore machine states such as register values.

need to save and restore Stack and register value

9. ([✓ T] / F) *In UNIX-like systems, fork() is a system call to create a new process.*

10. (T / [✓ F]) All threads in the same process share the same program counter.

11. ([✓ T] / F) User-level threads maintain thread control blocks in the user space.

User-level threads, the thread control block (TCB) is maintained in user space and not in kernel space

12. ([✓ T] / F) With virtual memory, instructions executed by the CPU use virtual addresses to locate data in physical memory.

13. ([✓ T] / F) MMU is a must requirement for modern virtual memory subsystems.

14. ([✓ T] / F) Paging-based virtual memory helps mitigate external fragmentation.

The solution to external fragmentation is compaction and paging.

15. (T / [✓ F]) Preemptive scheduling is always better than non-preemptive scheduling.

16. (T / [✓ F]) User-level threads **cannot** implement preemptive scheduling.

preemptive scheduling by using a timer interrupt mechanism or by yielding control to another thread voluntarily.

II. Single choice (3 points each) 12

17. Which of the following is correct about virtual memory with paging?

A. Solves the internal fragmentation problem paging solves external fragmentation and best-fit solves internal

B. Makes sharing of data between different processes harder

[✓ C. Offers protection domains] each process is isolated in its own virtual address space

D. Improves cache hit ratio

[✓ E. None of the above]

18. Consider a system using 32-bit address space, 1MB (=2^20 bytes) page size. Page table entry (PTE) is 4 bytes each. If single-level paging is used, what is the size of the page table for each process?

$2^{32} \times 2^{-20} \times 2^{2}$

12+2 = 14

A. 4096 bytes (= 2^12)

[✓ B. 16384 bytes (= 2^14)]

C. 262144 bytes (= 2^18)

D. 1048576 bytes (= 2^20)

E. 16777216 bytes (= 2^24)

19. Which of the following is correct about the Translation Lookaside Buffer (TLB)?

A. Delays large copies of memory as long as possible in fork()

B. Effective for write memory requests but not for reads

C. Performance is worse than caching PTEs in L1 cache

[✓ D. TLB typically has a high hit ratio due to locality] -

E. TLB often is the largest cache in the system -

20. Which of the following is correct about OS extensibility and protection?

A. DOS-like OSs provide good protection

B. DOS-like OSs have more border crossings than monolithic kernels

C. Monolithic kernels have more border crossings than microkernels

[✓ D. Monolithic kernels are bad at extensibility]

E. Microkernels are bad at extensibility

21. Which of the following is **NOT** correct about priority-based scheduling?

[✓ A. Need to know CPU burst time in advance] -

B. Priority assignment schemes are often ad hoc

C. Hard to achieve fair-share scheduling ?

D. Subject to the priority inversion problem

E. Effective in favoring important tasks over others

Justification of D This is a situation where a lower priority process holds a resource that a higher priority process needs, causing the high priority process to be blocked and not able to run

X 22. Which of the following is correct about multiprocessor scheduling? ↳ global

[✓ A.] Partitioned scheduling (aka. ~~single~~ multi queue multiprocessor scheduling) causes higher task migration overhead than global scheduling (aka. ~~multi~~ single queue multiprocessor sched.)

B. Partitioned scheduling is better at reclaiming unused processor time ×

C. Global scheduling tends to achieve better cache affinity ×

D. Push migration and pull migration cannot be used together in the same OS ×

[✓ E.] None of the above

III. Single choice/short answers (4 points each) 20.

23. Which of the following apply to the SPIN operating system?

A. No longer relies on virtual memory to enforce protection

B. Extensions are written in Modula-3

C. Capabilities are implemented directly through the use of pointers

D. Garbage collection may occur at runtime

[✓ E.] All apply to SPIN

24. Which of the following apply to the Exokernel?

A. Exokernel runs library OSs in the kernel space

[✓ B.] Downloaded code can reduce border crossings

C. *Visible resource revocation gives better efficiency as revocations happen more frequently*

D. *No intervention from Exokernel is needed when traps, faults, or interrupts occur*

E. *All apply to Exokernel*

25. Which of the following is **NOT** L4 microkernel's claim?

A. L4 Microkernel refutes the argument that "Microkernels are inherently slow".

B. The high overhead of previous microkernels (e.g., Mach) is an implementation issue

[✓ C.] As a microkernel is made more portable, its efficiency is likely to be improved. –

D. Major performance issues like switching overhead can be greatly reduced if the implementation considers architecture-specific features

E. All are correct.

26. Which of the following is correct about the lottery scheduler?

A. Deterministically fair scheduling ×

B. Good at achieving short-term fairness

C. Scheduler overhead is high

[✓ D.] Provides mechanisms to address the priority inversion problem

E. Ticket inflation leads to starvation

27. Which of the following is **NOT** correct about the Linux CFS scheduler?

A. Considers target latency to ensure response time
B. Considers minimum granularity to mitigate overhead
C. Makes a scheduling decision based on virtual time instead of physical time
[✓ D.] Uses a random function to achieve probabilistic-fair scheduling
E. All above are incorrect

29

IV. Short answers (5 points each) – Don't write long; one or two sentences should be enough.

28. Why the fork() system call is said to return "twice"? Explain briefly.

fork returns "twice". It returns child's PID to parent and 0 to the child.

5

29. What is the head-of-line blocking problem under FCFS scheduling? (**CEP question**) 5

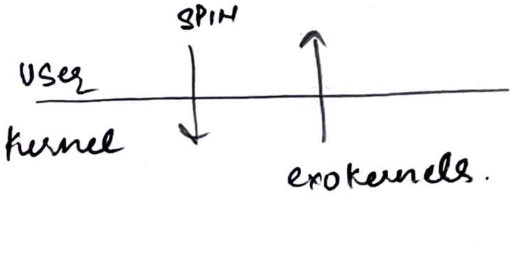
Head of the line blocking - is when long process can impede short process.
Eg:- CPU bound process followed by I/O bound processes

30. What is the head-of-line blocking problem of FCFS scheduling? (**CEP question**) 5

Same as 29

31. Briefly compare main design differences between SPIN and Exokernel. One sentence for each should suffice. (**CEP question**) 5

SPIN - adds application specific functionality in kernel.
Exokernel - make the barrier as low as possible.



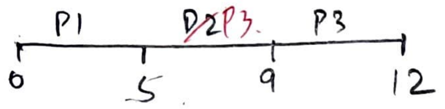
32. Consider a system running the following three processes:

Process	Burst Time	Arrival Time
P1	5	0
P2	4	1
P3	3	2

4

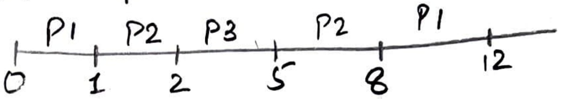
Compute the average turnaround time under (i) Non-preemptive Shortest Job First (SJF) and (ii) Preemptive SJF (PSJF) scheduling. Note: turnaround time = Completion time – Arrival time.

(i) Non-preemptive SJF



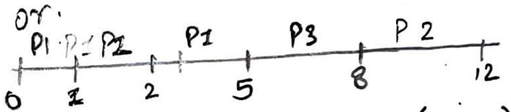
$$\frac{(5-0)+(9-1)+(12-2)}{3} = \frac{5+8+10}{3} = \frac{23}{3}$$ -1

(ii) Preemptive.



$$(12-0)+(8-1)+(5-2) = \frac{12+7+3}{3} = \boxed{\frac{22}{3}}$$

or.



$$\frac{(5-0)+(8-2)+(12-1)}{3} = \frac{5+6+11}{3}$$

$$\boxed{= \frac{22}{3}}$$

33. Consider a system with three processes:

Process	Arrival Time	Tickets
P1	0	5
P2	0	3
P3	0	2

Fill in the following table to show the scheduling decisions by Stride Scheduling. Assume 30 as the large constant (i.e., stride = 30 / number of tickets)

5

	Selected process	Pass values after selection		
		P1	P2	P3
Init	N/A	6	10	15
Time 1	P1	12	10	15
Time 2	P2	12	20	15
Time 3	P1	18	20	15
Time 4	P3	18	20	30
Time 5	P1	24	20	30
Time 6	P2	24	30	30

P1 → P2 → P1 → P3 → P1 → P2.

[End of document. Good luck!]