CS 111 Midterm

Stewart P Dulaney

TOTAL POINTS

100 / 100

QUESTION 1

1 Copy-on-write 8 / 8

Click here to replace this description.

✓ **+ 2 pts** default - copy on write - parent process's pages not copied for child process

✓ **+ 2 pts** if pages unmodified - they are shared between parent and child process

✓ **+ 2 pts** lazy creation of only modified pages per process

✓ **+ 2 pts** unnecessary data copying effort saved(time and memory)

+ 0 pts all wrong

QUESTION 2

2 Scheduling 10 / 10

✓ **+ 2 pts** largest overhead - round robin

✓ **+ 2 pts** round robin - why largest overhead

✓ **+ 1 pts** round robin - adv

✓ **+ 1 pts** round robin - disadv

✓ **+ 1 pts** fcfs - adv

✓ **+ 1 pts** fcfs - disadv

✓ **+ 1 pts** sjf - adv

✓ **+ 1 pts** sjf - disadv

+ 0 pts wrong answer

QUESTION 3

3 Running Processes 15 / 15

✓ **+ 3 pts** **Q1. X is running (A)**

✓ **+ 3 pts** **Q2. X is running (A)**

Q3.

+ 1.5 pts X is blocked (C) / Y is running (A)

✓ **+ 3 pts** **X is blocked (C) , Y is running (A)**

Q4.

+ 1 pts X is blocked (C) /Y is running (A) /Z is ready (B)

+ 2 pts Any two right from : X is blocked (C) /Y is running (A) /Z is ready (B)

✓ **+ 3 pts** **X is blocked (C), Y is running (A) , Z is ready (B)**

Q5.

+ 1 pts X is running (A) /Y is ready (B) /Z is ready (B)

+ 2 pts Any two from : X is running (A)/Y is ready (B) /Z is ready(B)

✓ **+ 3 pts** **X is running(A) , Y is ready(B) , Z is ready(B)**

QUESTION 4

4 fork() 16 / 16

✓ **- 0 pts** **(a) 3 times**

✓ **- 0 pts** **(b) 2**

- 8 pts Incorrect (a)

- 8 pts Incorrect (b)

QUESTION 5

5 Scheduling & turnaround time 18 / 18

✓ **- 0 pts** **Correct**

FIFO

- 3 pts Order incorrect

- 0.5 pts Turnaround calculation right (value wrong)

- 1.5 pts Turnaround calculation and value wrong

Round Robin

- 3 pts Order incorrect

- 0.5 pts Turnaround calculation right (value wrong)

- 1.5 pts Turnaround calculation and value wrong

SRTF

- 3 pts Order Incorrect

- 0.5 pts Turnaround calculation right (value wrong)

- 1.5 pts Turnaround calculation and value wrong

- 0.5 pts A and E messed up

Priority

- **- 3 pts** Order Incorrect
- **- 0.5 pts** Turnaround Calculation right (value wrong)
- **- 1.5 pts** Turnaround calculation and value wrong
- **- 0.5 pts** C and A messed up
- **- 1 pts** Did not find average (Divide by 5)

QUESTION 6

6 Clock Algorithm 16 / 16

- ✓ **- 0 pts Correct**
- **- 16 pts** Incorrect/ Not done
- **- 3 pts** Number of page faults not stated
- **- 4 pts** Not counting faults on startup
- **- 2 pts** 1 Load Error
- **- 4 pts** 2 Load Error
- **- 6 pts** 3 Load error
- **- 8 pts** 4 Load error
- **- 10 pts** 5 Load Error
- **- 12 pts** 6 Load Error
- **- 0 pts** Incomplete table
- **- 8 pts** No use bit
- **- 6 pts** Startup incorrect

QUESTION 7

ABI 17 pts

7.1 Windows & Solaris 4 / 4

- ✓ **+ 4 pts Emulation of Windows / Translator of ABI**
- **+ 2 pts** Mentions the relevance of ABI/OS
- **+ 1 pts** Thought-out reasoning
- **+ 0 pts** Incorrect
- **+ 0.5 pts** Some reasoning

7.2 Executing Programs 4 / 4

- ✓ **+ 4 pts Intercept calls and Simulate**
- **+ 2 pts** Thought-out Reasoning
- **+ 1 pts** Some Reasoning
- **+ 0 pts** Incorrect

7.3 Performance 3 / 3

- ✓ **+ 3 pts Reasonable assessment of tradeoffs**
- **+ 1.5 pts** Attempted assessment of tradeoffs
- **+ 0 pts** No attempt

7.4 Simulation 3 / 3

- ✓ **+ 1.5 pts Identify Problem**
- ✓ **+ 1.5 pts Identify Solution**
- **+ 0 pts** No attempt
- **+ 0 pts** Incorrect / Unclear Proposal

7.5 Solaris and SPARC 3 / 3

- ✓ **+ 3 pts Correct or Thorough + Reasonable Explanation**
- **+ 2 pts** Some flaws but Thorough Explanation
- **+ 1 pts** Flawed but Some explanation
- **+ 0 pts** No attempt
- **+ 0.5 pts** Only yes/no given

Midterm Examination
CS 111, Winter 2020
2/5/2020, 2 – 3:50pm

Name: Stewart Dulaney Student ID: 904-064-791

This is a closed book, closed notes test. One double-sided cheat sheet is allowed.

1. What is the benefit of using the copy-on-right optimization when performing a *fork* in the Linux system? **8 points**

You save processing time and memory by avoiding copying the data segment of the parent to a separate data segment for the child, potentially. If either writes to the data segment, the copy still occurs and the costs are only delayed.

2. Round Robin, First come First Serve, and Shortest Job First are three scheduling algorithms that can be used to schedule a CPU. What are their advantages and disadvantages? Which one is likely to have the largest overhead? Why? **10 points**

RR optimizes response time (RT) & fairness but has poor turnaround time (TAT). FCFS is simple and has low overhead but poor RT and TAT if long process runs first. SJF has better TAT than FCFS but poor RT for same reason as FCFS and also poor fairness. Also SJF assumes we know burst time which is not always true. RR would have largest overhead due to using preemption and expensive context switches.

3. Assume you have a system with three processes (X, Y, and Z) and a single CPU. Process X has the highest priority, process Z has the lowest, and Y is in the middle. Assume a priority-based scheduler (i.e., the scheduler runs the highest priority job, performing preemption as necessary). Processes can be in one of five states: RUNNING, READY, BLOCKED, not yet created, or terminated. Given the following cumulative timeline of process behavior, indicate the state the specified process is in AFTER that step, and all preceding steps, have taken place. Assume the scheduler has reacted to the specified workload change. **15 points**

For all questions in this Part, use the following options for each answer:
a. RUNNING
b. READY
c. BLOCKED
d. Process has not been created yet
e. Not enough information to determine

OR None of the above

(a) Process X is loaded into memory and begins; it is the only user-level process in the system. Process X is in which state?

X : a

(b) Process X calls fork() and creates Process Y. Process X is in which state?

X : a

Y : b

(c) The running process issues an I/O request to the disk. Process X is in which state? Process Y is in which state?

X : c

Y : a

(d) The running process calls fork() and creates process Z. Process X is in which state? Process Y is in which state? Process Z is in which state?

X: c

Y: a

Z: b

(e) The previously issued I/O request completes. Process X is in which state? Process Y is in which state? Process Z is in which state?

X : a

Y: b

Z: b

4. For the next two questions, assume the following code is compiled and run on a modern linux machine (assume any irrelevant details have been omitted): **16 points**

data

stack

code

```
main() {
   int a = 0;
   int rc = fork();
   a++;
   if (rc == 0) { rc = fork(); a++; }
   else { a++; }
   printf("Hello!\n");
   printf("a is %d\n", a);
}
```

Parent
a = 0 / 2

child
a = 0 / 2

child 2
a = 1 / 2

(a) Assuming fork() never fails, how many times will the message "Hello!\n" be displayed? 8 points

3 times for parent and child, the processes continue executing at the line after the fork() call. we have 3 processes (parent, child, child's child) that all run the printf("Hello!\n") line, since it's not wrapped in a conditional based on PID.

(b) What will be the largest value of "a" displayed by the program? 8 points

Parent child Child's Child

2 a = ϕ ✗ 2 a = ϕ ✗ 2 a = ✗ 2

a is a local variable stored in stack segment, when fork() the OS creates separate stack segment for child initialized to value of parents. As can be seen by my trace of a's value for each process, they all end up as 2 even though they execute different branches based on return code of fork().

5. **Scheduling.** Consider the following set of processes, with associated processing times and priorities:

Process Name	Processing Time	Priority
A	4	3
B	1	1
C	2	3
D	1	4
E	4	2

For each scheduling algorithm, fill in the table with the process that is running on the CPU (for time slice-based algorithms, assume a 1 unit time slice). **18 points**

Notes:
- A smaller priority number implies a higher priority.
- For RR and Priority, assume that an arriving thread is run at the beginning of its arrival time, if the scheduling policy allows it.
- All of the processes arrive at time 0 in the order Process A, B, C, D, E.
- Assume the currently running thread is not in the ready queue while it is running.
- Turnaround time is defined as the time a process takes to complete after it arrives
- SRTF, known as shortest remaining time first (SRTF), is another scheduling method that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute.

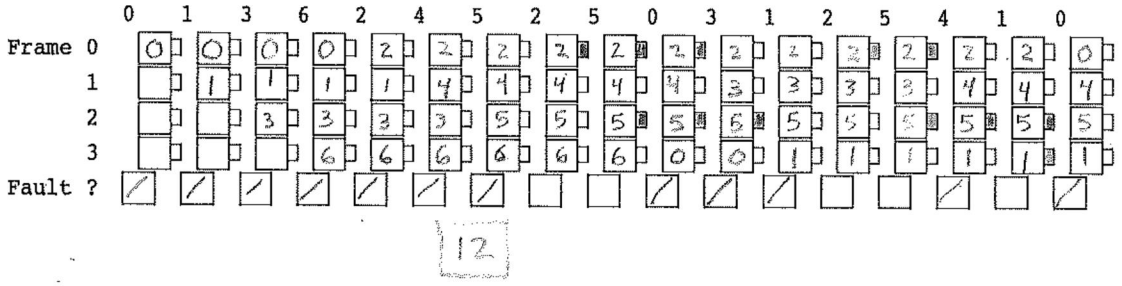
non-pre

Time	FIFO	Round Robin	SRTF	Priority
0	A	A	B	B
1	A	B	D	E
2	A	C	C	E
3	A	D	C	E
4	B	E	A	E
5	C	A	A	A
6	C	C	A	A
7	D	E	A	A
8	E	A	E	A
9	E	E	E	C
10	E	A	E	C

TAT	A	4-0=4	11	8	9
	B	5	2	1	1
	C	7	7	4	11
	D	8	4	2	12
	E	12	12	12	5
11		E	E	E	D
Average turnaround time		$\frac{9+15+12}{5} = \frac{36}{5}$	$\frac{13+11+12}{5} = \frac{36}{5}$	$\frac{9+6+12}{5} = \frac{27}{5}$	$\frac{10+23+5}{5} = \frac{38}{5}$

6. **Clock Algorithm**. The clock algorithm is an approximation of LRU based on using one use bit for each page. When a page is used its use bit is set to 1. We also use a pointer to the next victim, which is initialized to the first page/frame. When a page is loaded, it is set to point to the next frame. The list of pages is considered as a circular queue. When a page is considered for replacement, the use bit for the next victim page is examined. If it is zero [that page is replaced] otherwise [the use bit is set to zero, the next victim pointer is advanced, and the process repeated until a page is found with a zero use bit].

   Consider the reference string shown along the top of the following graphical structure. The system has four frames. Use the clock algorithm described in the previous paragraph. The narrow boxes to the right of the page number boxes can be used to keep up with use bits. Place the page number in the proper frame. Mark when page faults occur in the bottom line of boxes. State how many page faults occur. **16 points**



7. In the early 1990s, SUN Microsystems, the maker of the Solaris Operating System, wanted to move from the engineering desktop, where it was well established, to a broader market for personal productivity tools. The best personal productivity tools were all being written for Windows platforms, and SUN was on the wrong side of the applications/demand/volume cycle, which made getting those applications ported to Solaris a non-option. **17 points**

   One approach to their problem was to modify the version of Solaris that ran on x86 processors (the popular hardware platform for Windows) to be able to run Windows binaries without any alterations to those binaries. This would allow Sun to automatically offer all of the great applications that were available for Windows.

   (a) What would have to be done to permit Windows binaries to be loaded into memory and executed on a Solaris/x86 system? 4 points

   To support the ABI for Windows/x86 would need to support the Windows load module format and emulate Windows syscalls.

(b) What would have to be done to correctly execute the system calls that the Windows programs requested? 4 points

A new 2nd level trap handler would need to be written in order for Windows syscalls to work correctly after a trap and switch to kernel mode.

(c) How good might the performance of such a system be? Justify your answer. 3 points

It should be okay because only need to emulate syscalls not user level instructions. OS optimizes for spending as little time in kernel mode as possible which would help reduce performance costs.

(d) List another critical thing, besides supporting a new load module format and the basic system calls, that the system would have to be prepared to simulate? How might that be done? 3 points

Stack frame structure, parameter and return value passing, register conventions, DLL's, and more.
For DLL's, would need to emulate Windows Procedure Linkage Table and associated OS handlers. Context switching and conventions for what process state is saved would also need to be supported by the OS.

(e) Could a similar approach work on a Solaris/PowerPC or Solaris/SPARC system? Why or why not? 3 points

Yes if the ISA's matched. If they don't match, can run into issue's like 64 bit vs 32 bit word size. If ABI is well specified and interfaces followed by OS manufacturer, it should be possible.