Reminder

- 1st Midterm Exam
- Thursday February 18, 9.30am, in class
- Keep seat open between you and neighbor!
- Open book:
  - Textbook
  - Slides
  - Notes
  - If computer is used (e.g., slides) => no Wi-Fi, email, browser, chat, ...

Exam Question 1a

- Draw a diagram showing all possible process states and transitions between states.

![Diagram of process states and transitions](image)
Exam Question 1b

• Assume a process set A, B, C, D, E, F, and G, where A is currently being created, B is waiting for a disk operation to complete, C and G are currently executing, both D and E are waiting for execution, and F is a zombie process. Indicate in your diagram where these processes are situated at the current time (i.e., what states they are in).

Exam Question 1b

- A = NEW, B = WAITING, C = RUNNING, G = RUNNING, D = READY, E = READY, F = TERMINATED
Exam Question 1c

• Also explain in no more than one sentence how it is possible that two processes are executing at the same time.

• Multi-processor

Exam Question 2

• What is the difference between a child process right after a fork() system call and a child process after calling one of the exec() system calls?

• Fork(): “clone” of parent (identical content, but still separate process)
• Exec(): address space overwritten with new program
Exam Question 3

• 3. What does it mean when a process is “blocked” or “waiting”?

• In “blocked”/“waiting” state, waiting for some kind of event (e.g., I/O completion, timer going off, lock becoming available, etc.)

Exam Question 4

• You have written, compiled, and executed the following C program:

```c
#include ⟨stdio.h⟩
#include ⟨stdlib.h⟩
int main () {
    int pid1, pid2;
    pid1 = fork();
    if (pid1 == 0) {
        printf(“%d %d\n”, pid1, getpid());
    } else
        printf(“%d %d\n”, pid1, getpid());
}
```
Exam Question 4

- Assuming the fork() system call succeeded, specify the output of the program’s execution. Also assume that the parent’s process ID is 2125 and that of any child created will be greater than 2125 (e.g., 2126, 2127, ...).

- 0, 2126
- 2126, 2125

Exam Question 5

- Given the set of processes shown in the table below, draw the Gantt charts and compute the average waiting times for each of the following scheduling algorithms: (a) FCFS, (b) non-preemptive shortest-job-first, (c) preemptive shortest-job-first (also known as shortest-remaining-time-first), (d) round robin with q=4 (new processes join at the end of ready queue), and (e) preemptive priority scheduling. Which one has the shortest average waiting time? Is that consistent with your theoretical prediction? Justify your answer.
Exam Question 5

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Length</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>4</td>
<td>1 (Gold)</td>
</tr>
<tr>
<td>P2</td>
<td>4</td>
<td>3</td>
<td>3 (Bronze)</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>4</td>
<td>2 (Silver)</td>
</tr>
<tr>
<td>P4</td>
<td>6</td>
<td>2</td>
<td>2 (Silver)</td>
</tr>
<tr>
<td>P5</td>
<td>14</td>
<td>4</td>
<td>3 (Bronze)</td>
</tr>
<tr>
<td>P6</td>
<td>15</td>
<td>2</td>
<td>1 (Gold)</td>
</tr>
</tbody>
</table>

Exam Question 6

• A program consists of two processes shown below, where it is unknown in what order they will run. The dispatcher can switch between processes at any time. However, you can assume that no process will starve and that reading and writing single words (integers) are atomic operations. The initialization code is executed and completed before either of the two processes run. Both variables X and Y are shared between the two processes. Describe what will be the output when the program runs (if there are multiple possibilities, then describe all of them).
Exam Question 6

Initialization:
int X = 0; int Y = 0;

Process A:
while (X == 0) {
    // do nothing
}
printf(“a”);
Y = 1;
Y = 0;
printf(“d”);
Y = 1;

Process B:
printf(“b”);
X = 1;
while (Y == 0) {
    // do nothing
}
printf(“c”);

Exam Question 7

• What is the difference between preemptive and non-preemptive scheduling? Give an example task set (e.g., 2-3 tasks) and explain your answer using that task set (i.e., show the schedule).

• Preemptive: scheduler can decide to remove process from CPU; non-preemptive: process can only voluntarily give up CPU (yield, blocking, ...).

• Example: P1 (0,4), P2 (2,3); first schedule with FCFS, then schedule with RR and q = 3
Exam Question 8

- Name three items that a process control block (PCB) would want to track and why.
  - Process ID
  - Registers
  - Stack location/size
  - Heap location/size
  - Parents, siblings, children
  - Scheduling priorities
  - ...

Exam Question 9

- What does it mean for a system call to be blocking? How is it different from a non-blocking call?

  - Blocking: if it cannot be completed immediately; the process will “wait” (be placed in waiting queue, lose CPU, and be added to runqueue once reason for waiting is gone...)
Exam Question 10

• What is the difference between CPU-bound and I/O-bound tasks?

• CPU: process spends time primarily executing instructions (going back and forth between runqueue and CPU)
• I/O: process spends most time in waiting queue

Exam Question 11

• What is the purpose of a cache?

• Store frequently/recently used items closer to the CPU to increase memory access performance
Exam Question 12

- What makes it difficult to implement Shortest-Job-First in practice?

- Difficult to know/predict actual burst time.

Test-and-Set and Swap are two types of low-level operations that higher-level synchronization operations (e.g., semaphores) can be built upon.

True or False?
In RR scheduling, to keep scheduling overheads low, the time quantum should be small.

True or False?

When a child process is created, which of the following is a possibility in terms of the execution or address space of the child process?

- The child process runs concurrently with the parent.
- The child process has a new program loaded into it.
- The child is a duplicate of the parent.
- All of the above.
Which of the following scheduling algorithms is ALWAYS non-preemptive?

- SJF
- RR
- FCFS
- Priority algorithms