MTL458: Operating Systems
Quiz-1: Solutions

Duration: 30 minutes **Total: 10 marks**

Question 1: True or **False** with justification. Answers without a justification will be given **zero** marks.

1. If a child process terminates and its parent never calls `wait()`, will the child remain a zombie process permanently?

   – **False.** The child will remain a zombie as long as the parent is alive and does not call wait(). However, if the parent itself terminates, the zombie child will be re-parented to init (or systemd), which will call wait() and clean it up. Thus, the child does not remain a zombie permanently.

2. The Ready Queue stores processes that are waiting for I/O to finish.

   – **False.** The Ready Queue holds processes that are ready to run on the CPU; processes waiting for I/O are in a wait (blocked) queue.

3. A process undergoes a context switch every time it enters kernel mode from user mode.

   – **False**. After finishing its job in kernel mode, the OS may sometimes decide to go back to the user mode of the same process, without switching to another process.

[3*1=3 marks]

Question 2: Short answer questions (Unnecessarily long answer will be given zero marks)

1. How does fragmentation differ in pure segmentation and pure paging?

   – Segmentation: suffers from external fragmentation (free memory in scattered variable-sized holes). Paging: suffers from internal fragmentation (last page of a process may not be fully used).

2. How does MLFQ balance between responsiveness for interactive jobs and throughput for long jobs?

   – Interactive jobs stay in high-priority queues (short slices), long jobs drift down to low-priority queues (longer slices).

   * Interactive jobs get priority and quick response, while long jobs still make progress in lower queues with bigger quanta for good throughput.

3. What is the key difference between a mode switch (user $\leftrightarrow$ kernel) and a context switch?

   – Mode switch changes privilege level (user to kernel and vice versa) within the same process; context switch changes the running process itself.

   *Every context switch involves a mode switch, but not every mode switch is a context switch.

4. Consider a system with pure paging, using a single-level page table stored in main memory. (There is no TLB).

   Suppose the following instruction is stored at virtual address `1024`:

   ```
   1024 movl 200, %eax
   ```

   which loads the value stored at virtual memory address 200 into register `%eax`. How many memory references are required to execute this instruction? Justify!

[4*1=4 marks]

   – Instruction fetch-2 (1 to read the PTE for VA 1024), 1 to fetch the instruction). Operand fetch-2 (1 to read the PTE for VA 200, 1 to fetch the data at VA 200). Total, 4 memory references.

Question 3: A computer system uses **segmentation** with the following setup:

- Virtual address size: **14 bits**.
- The segment table for a process is as follows.

Segment	Base	Limit	Grow	Notes
00	4000	1000	1	Code (R/X)
01	8000	2000	1	Heap (R/W)
10	12000	1000	0	Stack (R/W)
11	–	–	–	Invalid

Translate the following virtual addresses into physical addresses or indicate if a segmentation fault occurs:

1. `10 000000001111`
2. `01 000100101100`
3. `00 010111011100`

[3*1=3 marks]

Solution

- Virtual address size: **14 bits**.
- Top **2 bits** = segment number (00, 01, 10, 11).
- Remaining **12 bits** = Offset within segment.

1. **Stack**
   Segment = 10, Grow bit is negative, Positive offset = 15 $\implies$ Negative offset = $15 - 2^{12}$ (since total size of each segment block = $2^{14}/4 = 2^{12}$), whose magnitude is above Limit = 1000 $\implies$ Segmentation Fault.

2. **Heap**
   Segment = 01, Grow bit is positive, Offset = 300. Limit = 2000, so valid.
   phys = $8000 + 300 = 8300$.

3. **Code**
   Segment = 00, Grow bit is positive, Offset = 1500. Limit = 1000, so *invalid.*
   $\Rightarrow$ Segmentation Fault.