Introduction

1. [3 points] Name the three main duties of the operating system?

Virtualization, concurrency, and persistence

2. [1 point] Why the operating system is called a resource manager?

OS allows resources to be shared among processes.

Processes

True or False
1. [0.25 point] System calls are interfaces between user programs and services that OS provides.
   True

True or False
2. [0.25 point] System calls can be run in either user mode or kernel mode.
   False

3. [0.25 point] The ………of a process contains temporary data such as function parameters, local variables, and return addresses.
   a) Code
   b) Heap
   c) Stack

4. [0.25 point] What data structure is used to hold information about a process when a context switch occurs?
   a) TLB
   b) Free list
   c) Process control block

5. [2 points] How many processes are created by the following program? Include the initial parent process. Assume that all of fork() functions run without errors.

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

int main() {
  for (int i = 0; i < 4; i++) {
    fork();
  }
  return 0;
}

16 processes

**Dynamic Relocation**

[2 points] Assume a process with an address space of 3KB (bound register) is loaded at the physical address 15KB (base register).

Using the dynamic relocation technique, compute physical addresses for the following virtual addresses. Explain your reason if a physical address cannot be computed.

<table>
<thead>
<tr>
<th>Virtual address</th>
<th>Physical address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1KB</td>
<td>16KB</td>
</tr>
<tr>
<td>4KB</td>
<td>Exceeds bound</td>
</tr>
</tbody>
</table>

**Segmentation**

[4 points] The following segment table shows the base and bound register pairs for each of code, heap, and stack segments in a 14-bit virtual address space.

Compute physical addresses for virtual addresses (in hex) below. Assume that the top two bits of the virtual address, 00, 01, and 11, indicate code, heap and stack segments, respectively. The maximum size of the stack segment is 4KB.

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>32KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Heap</td>
<td>34KB</td>
<td>2KB</td>
</tr>
<tr>
<td>Stack</td>
<td>28KB</td>
<td>2KB</td>
</tr>
</tbody>
</table>

a) 0x1068

0x1068: 01 0000 0110 1000 (14 bits)
01 (top two bits) → heap segment
Offset: 0000 0110 1000 (104)
34KB + 104B

b) 0x3C00
0x3C00: 11 1100 0000 0000 (14 bits)
11 (top two bits) → stack segment
Positive offset: 1100 0000 0000 (3KB)
Negative offset: 3KB (positive offset) – 4KB (maximum stack size) = -1KB
Physical address: 28KB (base address) + (-1KB) (negative offset) = 27KB

**Paging**

1. [1 point] Consider a virtual address space with a page size of 8KB. How many bits are used to represent the page offset in the address space?

   13 bits

2. [1 point] Given a virtual address 0xAEF9 (in hexadecimal) with a page size of 256 bytes, what is the offset (in hexadecimal digits)?

   a) 0xAE
   b) 0xF9
   c) 0xA
   d) 0xF900

3. [1 point] You are told that a given system has a 30-bit virtual address, with a 4KB page size. Assuming a page table entry size of 4 bytes, how big is a linear page table for a given process?

   a) $2^{20}$ bytes
   b) $2^{21}$ bytes
   c) $2^{22}$ bytes
   d) $2^{23}$ bytes

**Advanced Paging**

1. [1 point] Name the data structure that is accessed when a TLB-miss occurs.

   Page table is accessed.

   True/False

2. [2 points] Answer the following true/false questions about the multi-level page table.

   a) It is easier to allocate pages of the page table in a multi-level table compared to a linear page table.
b) Generally, multi-level page table lookups take longer than linear page table lookups.

True

**Swapping**

1. [3 points] Assume a memory that can hold 4 pages, and an LRU replacement policy. The first four references to the memory are to pages 6, 7, 7, 9.
   Assuming the next five accesses are to pages 7, 9, 0, 4, 9, which of those will hit in the memory?

<table>
<thead>
<tr>
<th>Page access</th>
<th>Memory state</th>
<th>Hit or miss?</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6, 9, 7</td>
<td>Hit</td>
</tr>
<tr>
<td>9</td>
<td>6, 7, 9</td>
<td>Hit</td>
</tr>
<tr>
<td>0</td>
<td>6, 7, 9, 0</td>
<td>Miss</td>
</tr>
<tr>
<td>4</td>
<td>7, 9, 0, 4</td>
<td>Miss</td>
</tr>
<tr>
<td>9</td>
<td>7, 0, 4, 9</td>
<td>Hit</td>
</tr>
</tbody>
</table>

2. True/False [2 points]
   Which of the following statements are true statements about various page replacement policies? More than one statement may be true.

   a) The optimal policy always performs at least as well as LRU. **True**
   b) When a page miss occurs, LRU replaces a page that was accessed the least number of times. **False**
   c) When a page hit occurs, FIFO changes the relative position of pages in memory to reflect the page hit. **False**
   d) LRU keeps track of the recency of pages with a time field in page tables. **True**

**Scheduling**

1. [4 points] The following table shows processes that each arrives to a system at some time and runs for a duration of time.

   a) What is the **average turnaround time** for FCFS and STCF schedulers? Assume that only STCF is preemptive.

   b) What is the **average response time** for a round robin scheduler with **time slice = 2**? Assume that while another job is running, a newly arrived job is placed on a FCFS ready queue.

   c) Assume that CPU is left idle for the 1st time unit. After that, **SJF** schedules jobs. What is the **average turnaround time** for SJF? Note that jobs P1 and P2 wait while CPU is idle.
<table>
<thead>
<tr>
<th>Job</th>
<th>Arrival time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.0</td>
<td>8</td>
</tr>
<tr>
<td>P2</td>
<td>0.4</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>1.0</td>
<td>1</td>
</tr>
</tbody>
</table>

a) FIFO: \( T_{\text{turnaround}} = ((8 - 0) + (12 - 0.4) + (13 - 1))/3 = 31.6/3 = 10.53 \)

STCF: \( T_{\text{turnaround}} = ((13 - 0) + (5.4 - 0.4) + (2 - 1))/3 = 19/3 = 6.33 \)

b) RR: \( T_{\text{response}} = ((0 - 0) + (2 - 0.4) + (4 - 1))/3 = 4.6/3 = 1.53 \)

c) SJF: \( T_{\text{turnaround}} = ((14 - 0) + (6 - 0.4) + (2 - 1))/3 = 20.6/3 = 6.86 \)

2. [1 point] One of rules in the MLFQ scheduler states that after some period of time, all jobs in a system are moved to the top-most queue.

What is the purpose of this rule?

Jobs do not starve on CPU time.

**Threads**

1. [2 points] Here is a multi-threaded code:

```c
#include <pthread.h>
#include <stdio.h>
void* mythread (void* arg) {
    long x = (long) arg;
    printf ("%ld", x);
    return NULL;
}

int main (int argc, char* argv[]) {
    pthread_t p[3];
    for (long i = 0; i < 3; i++) {
        pthread_create (&p[i], NULL, mythread, (void*) i);
    }
    for (long i = 0; i < 3; i++) {
        pthread_join (p[i], NULL);
    }
}
```

Assuming all of function calls succeed, select all possible outputs of this program from the following list:
2. [1 point] A thread-local storage refers to ..........portion of the address space of a process. This portion is owned by the thread.
   a) code
   b) stack
   c) heap

3. [2 points] The following program uses Pthreads API. What are the outputs of this program at LINE C and LINE P?

```c
#include <pthread.h>
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>

int value = 0;
void* worker (void* param) { //Thread
    value = 5;
    return NULL;
}

int main (int argc, char* argv[]) {
    pid_t pid;
    pthread_t tid;

    pid = fork();

    if (pid == 0) {
        pthread_create (&tid, NULL, worker, NULL);
        pthread_join (tid, NULL);
        printf (“child value: %d\n”, value); // LINE C
    } else if (pid > 0) {
        wait (NULL);
        printf (“parent value: %d\n”, value); // LINE P
    }
}
```

LINE C: 5
LINE P: 0