Graded Exercises

COMP-310/ECSE-427 Winter 23

Question 1: Forking [10 points]

1 – When a process makes a system call through the kernel API library with an address as argument, does the validity of that argument need to be checked in the library, in the kernel or in both places? Justify your answer. **[5 points]**

2 – How many times will the program below print `hello4`? Explain why (e.g., by drawing the process tree). **[5 points]**

```
main() {
  if (fork() !=0){
    if (fork () !=0)
      printf("hello1\n");
    else
      fork();
    printf("hello2\n");
  } else {
    if(fork() !=0) {
      printf("hello3\n");
    } else {
      fork();
    }
    printf("hello4\n");
  }
  printf("hello5\n");
}
```

Question 2: Synchronization [10 points]

Assume that a finite number of resources of a single type must be managed in a multi-process system. Processes ask for a number of these resources and –once finished– return them. The following program segment is used to manage a finite number of instances (`MAX_RESOURCES`) of an available resource. When a process wishes to obtain a number of resources, it invokes the `decrease_count` function. When a process wants to return a number of resources it calls the `increase_count` function.

```
#define MAX_RESOURCES 5
int available_resources = MAX_RESOURCES;

int decrease_count(int count){
  if (available_resources < count)
    return -1;
  else{
    available_resources -= count;
    return 0;
  }
}

int increase_count(int count){
  available_resources += count;
  return 0;
}
```

1 – Unfortunately, the program segment above produces a race condition. Identify the data involved in the race condition. **[2 points]**

2 – Identify the location (or locations) in the code where the race condition occurs. **[2 points]**

3 – Fix the race condition. You can either re-write pseudocode that modified the program above, or clearly explain how to modify the code. **[6 points]**

Question 3: Address Translation [10 points]

Given a CPU with 5-bit instructions, and 16 Bytes byte-addressable of physical memory, with the following 3 segments in main memory: SEG 0 (base address: 12_{10}, bound: 4_{10}), SEG 1 (base address: 8_{10}, bound: 1_{10}), SEG 2 (base address: 1_{10}, bound 3_{10}). X_{10} means X in base 10 (decimal).

Compute the virtual to physical address translations (or Segmentation Faults), for the following virtual addresses and explain your answers, e.g. by drawing a schema.
[2 points per address]

0_{10}, 7_{10}, 31_{10}, 20_{10}, 3_{10}.

Question 4: Compaction [10 points]

A memory management system eliminates holes by compaction.

Assume a random distribution of many holes and many data segments and a time to read or write a 32-bit memory word of 4 nanoseconds. For simplicity, assume that almost the entire memory has to be copied, because of the holes and data segment distribution.

1 – How long does it take to compact 4 GB **[5 points]**?
2 – Explain your reasoning **[5 points]**.

Question 5: Page Tables [10 points]

Assume that you have a machine with a 32-bit virtual address. The machine is paged, with a page size of 4 Kbytes, and a two-level page table scheme, with 1024 entries in the top-level page table.

1 – How many entries will there be in a single second-level page table? Justify your answer. **[4 points]**

2 – To represent a program with a sparse address space, with valid addresses from 0 to 18 Mbytes, and from 90 to 117 Mbytes, what is the minimum number of second-level page tables needed? Justify your answer. **[6 points]**