Chalmers University of Technology and Gothenburg University

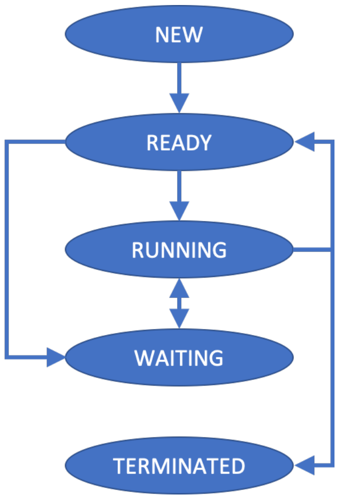
Operating Systems
EDA093, DIT401

Exam 2020-01-18

This exam has been conducted in Canvas.

Each question had several variations. This document reports only 1 variation per question.

(6 p) Indicate which connections are wrong and which ones are correct in the following figure. For each wrong or missing connection, explain why you think the connection is either wrong or missing.



Answer hint: please refer to the lecture about processes.

(6 p) Is the following code going to print “Greetings” or not? Explain why. Notice that the listing contains only the relevant portions of code, so your answer does not need to take minor inconsistencies (e.g., the missing include statements) into account.

```c
int main(void)
{
   char write_msg[BUFFER_SIZE] = "Greetings";
   char read_msg[BUFFER_SIZE];
   pid_t pid;
   int fd[2];

   /* create the pipe */
   if (pipe(fd) == -1) {
      fprintf(stderr,"Pipe failed");
      return 1;
   }

   /* now fork a child process */
   pid = fork();

    if (pid < 0) {
      fprintf(stderr, "Fork failed");
      return 1;
   }

   if (pid > 0) { /* parent process */
      /* close the unused end of the pipe */
      close(fd[READ_END]);
      /* write to the pipe */
      write(fd[WRITE_END], write_msg, strlen(write_msg)+1);
      /* close the write end of the pipe */
      close(fd[WRITE_END]);
   }
   else { /* child process */
      /* close the unused end of the pipe */
      close(fd[WRITE_END]);
      /* read from the pipe */
      read(fd[READ_END], read_msg, BUFFER_SIZE);
      printf("child read %s\n",read_msg);
      /* close the write end of the pipe */
      close(fd[READ_END]);
   }
   return 0;

}
```

Answer hint: yes, pipe before fork, and ends of pipe closed correctly before use.

(6 p) The price to parallelize an application whose code is 100% serial is of 30 dollars per percentage unit. That is, if you invest 60 dollars, you can get the serial part of the application to 98%. At the same time, if you want to run such an application on multiple cores, every extra core (aside for the first one) costs 50 dollars. What is better, to invest 210 dollars on parallelizing the code and 100 dollars in paying for cores or rather to invest 120 dollars on parallelizing the code and 200 dollars in paying for cores? Explain why in detail.

Answer hint: do the math using Amdhal's law.

(6 p) Given the following reference string:

$$7, 0, 1, 3, 1, 2, 0, 7, 4, 3, 2, 1, 0, 1, 7, 2, 0, 7, 1, 2, 3, 0, 1, 2, 7, 0, 7, 1, 0, 4$$

Assuming 3 frames (initially empty!) can be in memory for the process taken into account, and assuming the page replacement algorithm used by the OS is FIFO, show what is the page stored in each frame for each page request from the reference string and compute the total number of page faults.

Answer hint: solve has shown in the slides about VM.

(6 p) Write a short text (of at least 150 words) in which you explain why Virtual Memory, the Interrupt Vector Table and the type of Hard Disk used by a certain machine can **jointly** affect the performance experienced by an user (please notice: this question is about discussing the inter-connection between the 3 aspects, not simply about describing each aspect individually).

Answer hint: There can be several connections, of course. One can be that because of VM we have page faults. Page faults need to be solved rapidly, and IVT allows for fast reaction. Of course a page fault requires swapping pages, and there is a relation between type of disk and speed.

(6 p) Is the following code going to print the right value when executed passing parameter 10 or not? Explain why. If you think there is something to correct, point out all the corrections that must be applied to the code in order to work. Notice that the listing contains only the relevant portions of code, so your answer does not need to take minor inconsistencies (e.g., the missing include statements) into account.

```c
int sum;
void *runner(void *param);

int main(void)
{

   pthread_t tid;
   pthread_attr_t attr;

   if (argc != 2) {
       return -1;
   }

   if (atoi(argv[1]) < 0) {
       return -1;
   }

   pthread_attr_init(&attr);
   pthread_create(&tid,&attr,runner,argv[1]);

   printf("sum is %d\n",sum);

}

void *runner(void *param)
{

   int i, upper = atoi(param), sum = 0;

   for (i=1; i<=upper; i++)
    sum += i;

   pthread_exit(0);
}
```

Answer hint: Nope, no thread join and sum is locally created in runner.

(6 p) Describe how round-robin, priority and lottery scheduling work. Provide an example in which round-robin scheduling performs better than the other two.

Answer hint: please refer to the lecture about scheduling.

(6 p) Given the following table, and assuming Shortest Job First (non-preemptive), show how the CPU slots are allocated to processes and compute the average waiting time. All burst time are expressed in number or required CPU slots.

Process	Arrive Time	Burst Time
P1	0	10
P2	1	3
P3	3	5
P4	11	2
P5	12	3

Answer hint: P1 goes first. When P1 is over, P2 is chosen from P2,P3. When P2 is over, P4 is chosen from P3,P4,P5. When P4 is over, P5 is chosen from P3,P5. When P5 is over, P3 is chosen from P3... compute average waiting time based on this.

(6 p) Discuss why and how logical to physical address translation is done with O(1) time complexity with paging.

Answer hint: refer to slides from VM with translation of binary address. Fixed number of operations means O(1), so constant time.

(6 p) Can users use kernel threads? Motivate your answer.

Answer hint: Yes. Everything that is run by the OS is a kernel thread.