LI Operating Systems and Systems Programming

Solutions

Mock Exam January 2023

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 60, which will be rescaled to a mark out of 100.

Question 1

The question is about pointers and memory management in C.

(a) Will there be any memory leakage in the following program? Explain your answer.

```
1 int main()
2 {
3     int *A = (int *) malloc(sizeof(int));
4     scanf("%d", A);
5     int *B;
6     B = A;
7     free(B);
8     return 0;
9 }
```

[6 marks]

(b) A programmer has written the following function with the aim to return a pointer to an array of 10 random integers (`int`) to a caller function. There is a serious problem with this code. Explain what is wrong, why it is a problem, and how it can be fixed. Use this to write a correct version of the function without changing the function-signature. Assume that the caller function of `randomArray` is responsible for freeing any memory occupied by the array.

```
1 int* randomArray(void)
2 {
3     int array[10], i;
4     for (i = 0; i < 10; i++)
5        array[i] = rand();
6     return &array[0];
7 }
```

[7 marks]

(c) Consider the following two C functions `sum2Darray1` and `sum2Darray2`. Both of them compute the sum of all the elements of an input 2-dimensional matrix. Which one of them will be able to exploit memory hierarchy and thus achieve faster computation time? Explain your answer.

```
1 int sum2Darray1(int a[N][M])
2 {
3    int i, j, sum=0;
4    for(i=0;i<M;i++)
5      for(j=0;j<N;j++)
6        sum =sum + a[j][i];
7    return sum;
8 }
```

```
1 int sum2Darray2(int a[N][M])
2 {
3    int i, j, sum=0;
4    for(i=0;i<N;i++)
5      for(j=0;j<M;j++)
6        sum =sum + a[i][j];
7    return sum;
8 }
```

[7 marks]

Model answer / LOs / Creativity:

(a) The program will not leak memory.

The allocated memory-block is pointed by A and later by both A and B. Hence, both A and B contain the same address. free(B) is the same as free(A) and hence there will be no memory leak.

2 marks will be awarded if the student writes that the program will not leak memory. 4 marks will be awarded for the explanation.

(b) The array is allocated in the stack frame of randomArray(), hence it is a local object. The array object is only in scope during the execution of the function. At the completion of a randomArray() function call, the stack frame will be deallocated and the local array object will be out of scope.

If the function returns a pointer to an out-of-scope array object, then the pointed data will not be reliable and hence the program output will not be reliable.

Instead of creating a local array object in the stack frame, the function should allocate the array in the heap and then return a pointer to the heap-based array.

```
1 int* randomArray(void)
2 {
3     int *array, i;
```

```
4     array = (int *) malloc(10*sizeof(int));
5     for (i = 0; i < 10; ++i)
6        array[i] = rand();
7     return array;
8 }
```

For finding the problem, 3 marks will be awarded. For providing a solution, 4 marks will be awarded.

(c) Modern computers with memory hierarchy try to speedup computation by applying the principles of temporal and spatial locality. Thus, when a program tries to read one `int` object from the main memory, other adjacent `int` objects are also brought to the cache.

The function `sum2Darray1` reads the matrix elements along the columns. Whereas `sum2Darray2` reads the matrix elements along the rows. Since, the C programming language stores a 2D matrix in the row-major order, `sum2Darray2` offers better spatial locality compared to `sum2Darray1`, and thus offers better performance.

Hence, Strategy2 will be more efficient

2 marks for identifying the fastest strategy and 5 marks for the explanation.

Question 2

(a) The question is about Main and Virtual Memory. Provide a brief answer.

(i) Why does a processor have a set of registers in addition to a large main memory? **[1 mark]**

(ii) A scheduler controls the degree of multi-programming in an Operating System. The scheduler can send a process to which state(s)? **[1 mark]**

(iii) Does adding more frames during Page Replacement always lead to improved performance? **[1 mark]**

(iv) A system is running with following measure behaviour: CPU utilization 10%; Paging disk 95% Other I/O devices 3%. Explain which of the following actions will improve CPU utilization and why?

i. Install more main memory

ii. Install a faster disk

iii. Changing the degree of multi programming **[3 marks]**

(b) Briefly describe what the possible consequences are of a buffer overflow in the kernel. **[6 marks]**

(c) Consider the following piece of kernel code. The intention is that whenever data is written to a proc-file, this data is written to a device. The device provides two functions: **start_transfer** starts the transfer of count bytes to the device and returns immediately, and **transfer_finished**, which is called by the device when the data transfer is finished. The function **kernelWrite** should return the number of bytes transferred to the device.

```
 1 int total_transferred = 0;
 2 /* total number of bytes transferred since module loaded */
 3 int transferred = 0;
 4 /* bytes transferred to device in single transaction */
 5
 6 /* called by device when transfer finished */
 7 /* called in interrupt mode */
 8 void transfer_finished(int count) {
 9   transferred = transferred + count;
10   /* wakeup waiting process */
11 }
12
13 /* called every time data is transferred to kern
14     ,as a result of writing to proc-file  */
15 int kernelWrite(char * buffer, int count) {
16   /* buffer is pointer to user space */
17   start_transfer(buffer, count);
18   /* go to sleep until woken up in transfer finish */
19   transferred = transferred + count;
20   return transferred;
21 }
22
23 void init module(void) {
24   /* set up proc-structure - code omitted */
25   proc->write_proc = kernelWrite;
26 }
27 }
```

This kernel code compiles correctly but does not work as intended. Identify these errors and suggest remedies. If you think critical sections are required, it is sufficient to indicate begin and end of a critical section, and whether you would use semaphores or spinlocks to protect the critical section.

[8 marks]

Model answer / LOs / Creativity:

(a) (i) Registers are normally at the top of the memory hierarchy, and provide the fastest way to access data.

(ii) Ready, suspend

(iii) No. Adding more frames may decrease performance. Belady's Anomaly.

(iv) All three actions will improve CPU utilization.
-With more main memory, more pages can stay in main memory and less paging to or from the disk is required.
-Faster disk means faster response and more throughput, so the CPU will get data more quickly.
-Decreasing the degree of multi-programming will help.

Marking: 1 marks for each question.

(b) Any answer that would correctly describe that arbitrary data may be overwritten, hence any process or even the kernel may be corrupted and crash. **Marking: 3 marks for the reason and 3 marks for how might affect the kernel buffer**

(c) The errors and fixes are:

- The variable `transferred` is increased twice, whereas the variable `total_transferred` is not increased at all. Each variable must be increased once within a critical section. The variable `transferred` should be declared within the `kernelWrite` function and initialised to 0. If a critical section happens wthin `transfer_finished`, spinlocks must be used to protect this critical section. If a critical section happens within `kernelWrite`, semaphores should be used.
- The data must be copied from `buffer`, which is in user space, to a buffer in kernel space eg. via `copy_from_user`.
- The function `kernelWrite` needs to return the number of bytes transferred. In the proposed solution line 20 happens to be OK but this needs to be checked for other solutions.

Other solutions are OK as long as they address the problems.

Question 3

(a) Four processes running on a single-core processor (Table 1). Among all the Scheduling Algorithms, briefly explain which one you will prefer and which one you would like to avoid for the given scenario? **[5 marks]**

(b) Predict all possible outputs that the following C program will print to the console and briefly explain your answer. What will be the state of parent process? Briefly explain the behaviour of the program if we comment out the line number 16.

Table 1: List of Processes

Process	Type	Arrival time	Burst time
P1	CPU Bound	0	50
P2	I/O Bound	1	3
P3	I/O Bound	2	4
P4	CPU Bound	3	20

```
 1 #include <sys/types.h>
 2 #include <sys/wait.h>
 3 #include <unistd.h>
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6 int main() {
 7    pid_t   wpid, child_pid;
 8    int status = 0;
 9    int i;
10    for(i = 0; i < 2; i++) {
11        if ((child_pid = fork()) == 0) {
12           printf("process \n");
13           exit(0);
14        }
15    }
16    while ((wpid = wait(&status)) > 0);
17    return 0;
18 }
```

[5 marks]

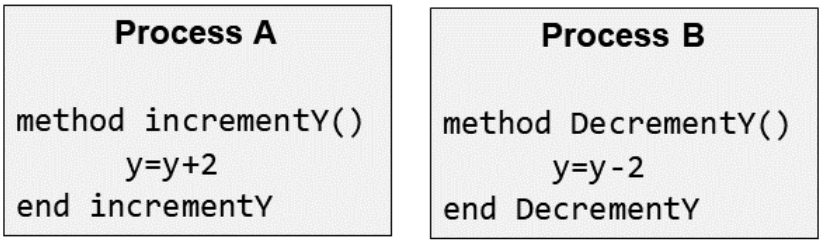
(c) Your computer system uses Round Robin scheduler and is not very responsive and so you decide to change the scheduling time quantum from 50 msec to 1 msec. Now the performance is even worse. Why is this happening? **[4 marks]**

(d) Consider a concurrent system with two processes A and B (Figure 1). Assume y is a shared variable with value of 5. Describe how a race condition is possible and provide a solution to prevent the race condition from occurring. **[6 marks]**

Model answer / LOs / Creativity:

(a) We should avoid First Come, First Served (FCFS) Scheduling because processes with short burst time may have to wait for processes with long burst time.
We can use either Shortest Job First (SJF) Scheduling, Shortest remaining time first (SRTF), Round Robin (RR). **Marking: 2 marks for identifying algorithms to avoid and 3 mark for proposing SJF, SRTF, RR.**

Figure 1: Concurrent processes A and B



(b) If the fork() succeeds, Line 12 will be printed twice. If the fork() fails, then no output will be generated. Parent process will be in wait state.
If we comment out line 16, the parent will not wait for the child process to complete (Zombie process).
Marking: 2 marks for fork() success. 2 marks fork() failure and 1 marks for zombie process.

(c) Any correct answer that would describe that extra context switches are taking place
Marking: 4 marks for correct answer.

(d) The assignment operation for y can be interrupted while execution leading to values different than actual value 5 after calling the method increment and decrement in a sequence.
Proposed solution (Figure 2).: Make shared variables y and z atomic. Use any lock technique (mutex, semaphores, hardware lock etc). **Marking: 3 marks for identifying the issue. 3 marks for proposing the solution**

Figure 2: Concurrent processes A and B with locks