Operating Systems - Monsoon 2024

September 8, 2024

Assignment 1 (Total points: 100)
Due: September 15, 2024

Instructions.
1. Follow all the instructions in the questions closely.
2. Code should be written in C language.
3. Files should be named q[question number].c unless stated otherwise. For example: q1.c
4. Submit a .zip named [RollNo].zip file containing source code files and write-up.
5. During the Assignment demos, there will be questions on the code and its working. Ensure that you know the code you are submitting.
6. Assignment need to be worked on individually. This is not a group assignment.
7. After the deadline, for every 6 hrs, 5% of points will be cut.

Q1. Write a program where the parent process has 7 children. Each child generates 4 random numbers from 1 to 100, calculates their mean, prints it in a new line and returns. The parent should wait for the children to finish and then return. **[10 pts]**
Output: 7 random number means calculated by each child.

Rubrics:
The student should have implemented the following things:
- File q1.c containing the main function.
- The parent process should fork 7 times, possibly in a for loop.
- Each child should pick 4 random numbers, average them, print the average.
- The parent should call wait() 7 times to wait for each of the children.
- The printed output should have 7 random means printed one after the other.

Q2. Perform a binary search on a 16-element sorted array by passing half the array to a child process. The parent should define the array, print it and ask the user to input a target value. The parent will compare the target value to the mid element, call fork, and and the child will operate on half of the array. Similarly, the next child will operate on ¼ of the array and so on. The child process that finds the target value should print its index in the terminal. If the target doesn't exist in the array, print -1. Each parent should wait for their child to finish before returning. **[25 pts]**
Input: Define a 16-element array in the code. The array must be sorted.
Output: Output format should be as follows:

```
Array: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Enter a target value: 6
```

```
Target value's index: 5
```

Rubrics:
The student should have implemented the following things:

- File q2.c containing the main function.
- Defined a sorted 16-element array.
- Asked user for target value.
- A child should operate on half of the array of its parent at each fork level. There will be, at most, 4 forking levels.
- The correct index of the target value in the array is printed. If the target is not in the array, -1 is printed.

Q3. For this question, create a folder `q3` which should contain the following files:

1. **date.c:** Contains a simple implementation of the `date` command. This program should print the current date in at least three different formats. The user can pass options/flags to specify the desired format (e.g., `-u` for UTC, `-r` for RFC 2822 format). If no flag is passed, it should print the default date and time format, similar to the linux *date* command.
2. **cal.c:** Contains a simple implementation of the `cal` command using Zeller's congruence. The program should take month and year as arguments and calculate the first day of the month using the formula for the Gregorian calendar. If the month and year are not provided, it should print an error message. Use Zeller's congruence to determine the first day of the given month and construct the rest of the month.
3. **uptime.c:** Contains a simple implementation of the `uptime` command. This program should print the system's uptime using the `sysinfo()` function. If no additional options are provided, print the uptime in hours, minutes, and seconds. Add an error message if there are any issues retrieving the uptime.
4. **main.c:** This program will run all the executables of the above programs as child processes using the fork-wait-exec sequence. First, create 3 child processes using `fork`. Then, inside the child processes, call one of the `exec` functions to execute the above programs. Each child will run one program. The parent process will wait for all child processes to finish before returning.
5. **Makefile:** This makefile should build the first 3 programs (`date`, `cal`, `uptime`) before building the `main` program. Your files should be correctly compiled using this Makefile. The output of compiling a C file should be an executable file of the same name. For example: `date.c` should be compiled to `date` in Linux. **[40 pts]**

Rubrics:

The student should have implemented the following things:

- Folder q3 containing the files `date.c`, `cal.c`, `uptime.c`, `main.c`, and `Makefile`.
- All files should be correctly compiled without errors using `make`.
- In `main.c`, the parent process must use `fork()` 3 times, then `wait()` 3 times for each of the children to finish. The children should call an `exec` function on `./date`, `./cal`, and `./uptime`, respectively.
- Running the `main` executable should:
    - Print the current date and time in the default format using the `date` program.
    - Days of the month should be printed using the *cal* program, similar to the cal output on Linux based systems.
    - Show the system's uptime using the `uptime` program.

Q4. Write a c program to simulate the different CPU scheduling algorithms. Four scheduling algorithms should be implemented in the program: **[25 pts]**

1. First In First Out (FIFO)
2. Shortest Job First (SJF)
3. Shortest Remaining Time First (SRTF)
4. Round Robin (RR)

The program should accept input for n processes (n>4), where each process has:

- **Process ID**: A unique identifier for the process.
- **Arrival Time**: The time at which the process arrives in the ready queue.
- **Burst Time**: The total time required by the process for execution.
- **Time Quantum (TQ)**: The value of TQ should only be used in RR.

For each scheduling algorithm, the program should:

- Calculate and display the **order of execution** of processes.
- Calculate and display the **scheduling policy's Average Response Time** and **Average Turnaround Time**.

Rubrics:
The student should have implemented the following things:
● Program source code in .c extension.
● The output should contain the order of execution of the processes and the Average response time and average turnaround time for each scheduling algorithm.
● fork(), wait(), sleep(), kill() these functions can be used.
● Error handling in terms of wrong command, wrong option or wrong argument