Operating Systems - Monsoon 2023

Dhruv Kumar
August 19, 2023

Assignment 1 (Total points: 100)
Due: Aug 29, 2023. Time: 23:59 Hrs. (Hard Deadline)

Instructions-
There should be ONLY ONE submission per group
Submit a *.zip* named RollNo1_RollNo2.zip file containing code and write-up.

Updates after the initial posting of assignment are highlighted in yellow.

Q1 : Create a parent-child relationship between two processes. The parent should print two
statements: **(30 Marks)**

1) A) Parent (P) is having ID <**PID**>
   B) ID of P's Child is <**PID_of_Child**>
   The child should print two statements:
   C) Child is having ID <**PID**>
   D) My Parent ID is <**PID_of_Parent**>

Make use of wait() in such a manner that the order of the four statements A, B, C and D is:
A, C, D, B. You are free to use any other relevant statement/printf as you desire and their order
of execution does not matter.

2) Write a program to create two processes using vfork() system call in which the
   child process will calculate the factorial of 4, and the parent process will
   calculate the Fibonacci series up to 16. Parent should wait for the child to
   complete its working.
   **Clarification: For Fibonacci series, you have to calculate and print the first 16
   elements of Fibonacci series**

   The following ques tion is a bonus part:

   Write a program to create two processes using **vfork()/fork()** system call in
   which the child process will calculate the factorial of 4, and parent process will

Rubrics:

Student should have implemented following things:
- Program source code(s) with Makefile to compile.
- Write-up giving a brief description of how the program works (less than 1 page)
- Error handling

Q2 : You're making a special unix system, and your project manager wants you to create three specific commands for c shell (c program that acts as a shell for your system) to be used in it.**(50 Marks)**

1) **word:** It is a built-in **(internal)** command, Reads the number of words **(assume word is a "space-separated" string)** in a text file, and throws an error if the file does not exist.
Syntax: word [-option] [file_name]
Note: if no option is passed then there should be only one file, the number of files may vary as per option and the same would reflect in syntax .
It should additionally also cater 2 options:
- -n : ignores new line character
- -d : difference between the word sizes of two text files

Note: Only one of the options (-n or -d) can be used at a time with the **word** command.

2) **dir:** It creates a directory, and then changes the path to that directory. It is an external command, throw an error if that directory already exists.
Syntax: dir [-option] [dir_name]
It should additionally also cater 2 options:
- -r : removes if the directory already exists and create a new directory instead of throwing an error
- -v : print a message for each step in the running of this command

3) **date:** It returns the last modified date and time of a particular file. It is an external command
Syntax: date [-option] [file_name]
It should additionally also cater 2 options:
- -d : display time described by STRING
- -R : output date and time in RFC 5322 format

Rubrics:

Student should have implemented following things:
- Program source code(s) with Makefile to compile.
- Write-up giving a brief description of how the program works (less than 1 page)

Q3: Make an arithmetic calculator for a school. Follow the given instructions for the same. **(20 Marks)**

Create a bash script shell that acts like a math calculator. It should do these things:

1) Read a text file named "**input.txt**" that has two numbers and an operation in the format **x y operation** where x and y are numbers and the operation is the name of the command. **Note:** The file may have more than one set of inputs.
2) Calculate the result of that operation.
3) Save the result in a new text file named "**output.txt**" in the directory named "**Result**" (if the directory doesn't exist in the current directory, make it).

   There are only three operations:
   1) "**xor**": Get the xor of the two given numbers.
   2) "**product**": Get the product of the two given numbers.
   3) "**compare**": Get the bigger number from the two given numbers.

Test Cases for the Assignment - 1

Q1) There is no test case for this question, the flow of expected output is mentioned in all three parts, and your solution should give the output in the same flow.

Q2)

Note: The following output is based on my assumptions for the approach, your output may vary but you have to mention your assumptions in the documentation clearly. The given output is just for the reference to your understanding of commands.
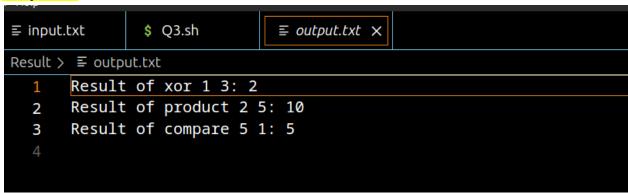
```
shell ~/Desktop >> word input.txt
7
shell ~/Desktop >> word -n input.txt
4
shell ~/Desktop >> word -d 1.txt 2.txt
2
shell ~/Desktop >> word -t input.txt
Error: The command is incorrect
shell ~/Desktop >> word input1.txt
Error: File doesnot exist
shell ~/Desktop >> date input.txt
Mon Aug 25 01:15:50 PM IST 2023
shell ~/Desktop >> date -R input.txt
Mon, 25 Aug 2023 13:15:50 +0530
shell ~/Desktop >> date -d 'yesterday' input.txt
Mon Aug 24 01:15:50 PM IST 2023
shell ~/Desktop >> date -t input.txt
Error: The command is incorrect
shell ~/Desktop >> date input1.txt
Error: File doesnot exist
shell ~/Desktop >> dir Basics
Error: Directory already exists
shell ~/Desktop >> dir -r Basics
shell ~/Desktop/Basics >> dir Basics1
shell ~/Desktop/Basics/Basics1 >> dir -v Basics2
Basics2 is created in ~/Desktop/Basics/Basics1
Directory is changed to ~/Desktop/Basics/Basics1/Basics2
shell ~/Desktop/Basics/Basics1/Basics2 >> dir -p 1
Error: The command is incorrect
shell ~/Desktop/Basics/Basics1/Basics2 >> cd 1
Error: The command is incorrect
shell ~/Desktop/Basics/Basics1/Basics2 >> exit
```
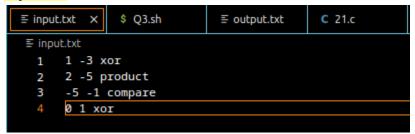
Q3)
Input.txt


```
≡ input.txt    X      $  Q3.sh           ≡ output.txt

  ≡ input.txt

    1      1 3 xor
    2      2 5 product
    3      5 1 compare
    4
```

Output.txt

```
≡ input.txt          $ Q3.sh          ≡ output.txt ✕

Result >  ≡ output.txt
   1      Result of xor 1 3: 2
   2      Result of product 2 5: 10
   3      Result of compare 5 1: 5
   4
```

Input.txt

```
≡ input.txt  ✕    $ Q3.sh        ≡ output.txt      C 21.c

  ≡ input.txt
   1      1 -3 xor
   2      2 -5 product
   3      -5 -1 compare
   4      0 1 xor
```

Output.txt

```
≡ input.txt        $ Q3.sh        ≡ output.txt ✕    C 21.c

Result >  ≡ output.txt
   1      Result of xor 1 -3: -4
   2      Result of product 2 -5: -10
   3      Result of compare -5 -1: -1
   4      Result of xor 0 1: 1
```