**Read the following to understand the parent & child process concept.**

* Process state Diagram
* The parent & child process concept
* Understand orphan processes
* Understand zombie processes
* what is **getpid()** and how to use it..??
* what is **getppid()** and how to use it..??
* what is **fork()**? what happens when you use fork() in your program..??
* what is **wait()** / **waitpid()** and how to use it..??
* what are **zombie processes**? when they will get created.? How to kill the zombie processes..?
* what are **orphan ​processes**? when they will get created.?
* what is user space?
* what is kernel space / system space?
* What is **IPC**(inter process communication) and what is the need of it.
* what is **Unnamed PIPE?** how to create it? When we need this??
* what is **named PIPE**? how to create it? When we need this??
* What is the difference between **Unnamed PIPE** and **named PIPE**?
* what are execl,exec,execv and difference between them..??
* what is **semaphore**? Explore sem\_init(), sem\_create(), sem\_destroy(), sem\_unlink(), sem\_wait(), sem\_post().
* what is **mutex**? Difference between semaphore and mutex.

**Process Management**

Note :

1. Read Man pages of the functions/commands mentioned in the hint of each question.
2. You will get the name of header file to be included for that function to work, in the man page of that respective function.

Questions :

1. Parent and child process

Write a C/CPP program to create a child process. Child should print its pid and its parent’s pid and should exit by printing message as “Child Exiting …”. Parent should print its pid and should exit by printing message as “Parent Exiting ..”.

(Hint(functions to be used) : getpid, fork, getppid)

1. Scheduling functions

Write a program in C/CPP to check the scheduling policy used by the process and its priority.

(Hint(functions to be used) : sched\_getscheduler, getpid)

1. Scheduling functions

Write a program in C/CPP to get the current scheduling policy of the process. The program should change the scheduling policy to the other than current one. Program should report errors if it fails to set the new scheduling policy.

(Hint(functions to be used) : sched\_setscheduler, getpid)

1. Scheduling algorithm

Write a program in C/CPP to take process name, its arrival time and execution/burst time as input.

Use FCFS(non-preemptive) algorithm to calculate wait time of each process, average

wait time, turnaround time of each process and average turnaround time.

Calculation of time will start from the arrival time of first process.

execution/burst time - Time required for execution of process

Wait time of process = response time of process - arrival time process

Response time of process : time at which process is scheduled to run

Average wait time = (sum of wait time of each process) / (number of processes)

Turnaround time of process = (finish/completion time of process) - (arrival time of process)

Average turnaround time = (sum of turnaround time of each process) / (number of processes)

Sample Input

|  |  |  |
| --- | --- | --- |
| Process | arrival time | execution/burst time |
| P1 | 0 | 3 |
| P2 | 2 | 5 |
| P3 | 5 | 6 |

Sample Output

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | Response time | Completion/finish time | Waiting time | Avg waiting time | Turnaround time | Avg turnaround time |
| P1 | 0 | 3 | 0-0 = 0 | (0+1+3)/3=4/3 | 3-0=3 | (3+6+9)/3=18/3=6 |
| P2 | 3 | 8 | 3-2=1 | 8-2=6 |
| P3 | 8 | 14 | 8-5=3 | 14-5=9 |

Write a C/CPP program to create a child process. Child should print its pid and its parent’s pid and should exit by printing message as “Child Exiting …”. Parent should print its pid and should exit by printing message as “Parent Exiting ..”. sudo apt install gcc

-----

#include<iostream>#include<sys/types.h>#include<unistd.h>using namespace std;int main(){pid\_t pid;pid=fork();if (pid==0){cout<<"this is child process:"<<getpid();}else if (pid >0){cout<<"this is parent process:"<<getpid();}else{cout<<"process is not created";}}

Create 4 sibling processes each doing one of the tasks given below. Sibling 1 : - lists files or directories from the current directory. Sibling 2 : - displays the contents of the program file. Sibling 3 : - displays today’s date. Sibling 4 : -execute the bash command. The main parent process should wait for all four child processes to finish. It then prints “Goodbye” before terminating. (Hint: Use execl. zombies and orphans must not be formed) for example, if you are using execl(), following are the key points: ret = execl(param1,param2,param3,param4,param5,....paramn,NULL); - param1 must be the absolute pathname of the utilities / application - param2 is quite simple - just pass the name of the binary or application - from param3 to paramn, number of parameters and the value passed for The parameters is decided by the binary/application - you must end execl() with a NULL this means, no more parameters are present

From vineeta singh to Everyone: 06:09 PM

<https://www.onlinegdb.com/online_c++_compiler>

Write a C++ program to create a new string with the last char added at the front and back of a given string of length 1 or more. "Red""Green""1"Sample Output:dReddnGreenn111###################################### execl("/bin/date", "date",0);