|  |
| --- |
| **Lab: 3**  **Task 1: write a program to print the process Id,**  **Task 2: write a program to implement fork(), getpid(), getppid(), exit() system call.**  **Task 3: write a program to implement FCFS scheduling algorithm and analysis the result** |

1. **Program to print the process id**

|  |
| --- |
| #include<stdio.h>  #include<unistd.h>  int main()  {  int pid,ppid;  pid=getpid();  ppid=getppid();  printf("\n Process Id is %d\n",pid);  printf("\n Parent Process Id is %d\n",ppid);  return 0;  }  **OUTPUT**  process ID is 5198  parent process ID is 5129 |

1. *System calls (fork, getpid, getppid, exit)*

*Aim*

*To create a new process with appropriate system calls.*

***Algorithm***

*1. Create a new process by invoking the system call* ***fork*** *(the return code for the fork system call is zero for the new (child) process, whereas the nonzero process identifier of the child is returned to the parent.*

*2. Get the process identifier of the currently running process and its respective parent using the system calls:* ***getpid*** *and* ***getppid.***

*3. Display the same.*

***Program***

|  |
| --- |
| *#include<unistd.h>*  *#include<stdlib.h>*  *#include<stdio.h>*  *int main()*  *{*  *int pid;*  *printf("Process related system calls");*  *if((pid=fork())==-1)*  *{*  *printf("Error During fork");*  *exit(0);*  *}*  *if(pid==0)*  *{*  *printf("child process");*  *printf("process id of child in child=%d",getpid());*  *printf("process id of parent in child=%d",getppid());*  *exit(0);*  *}*  *else*  *{*  *printf("parent process");*  *printf("process id of child in parent=%d",getpid());*  *printf("process id of parent in parent=%d",getppid());*  *exit(0);*  *}*  *return 0;*  *}* |

|  |
| --- |
| *#include<unistd.h>*  *#include<stdlib.h>*  *#include<stdio.h>*  *int main()*  *{*  *int pid;*  *printf("Process related system calls");*  *if((pid=fork())==-1)*  *{*  *perror("During fork");*  *exit(0);*  *}*  *if(pid==0)*  *{*  *printf("child process");*  *printf("process id of child in child=%d",getpid()); printf("process id of parent in child=%d",getppid());*  *exit(0);*  *}*  *else*  *{*  *printf("parent process");*  *printf("process id of child in parent=%d",getpid()); printf("process id of parent in parent=%d",getppid());*  *exit(0);*  *}*  *return 0;*  *}* |

***Sample 1/0***

*process id of child in child=1235 process id of parent in child=7645*

*fork() returns the value of the child’s PID to the parent process and zero to the child process.*

*#include <stdio.h>*

*main()*

*{*

*int ret;*

*ret = fork();*

*if (ret == 0)*

*{*

*/\* this is the child process \*/*

*printf("The child process ID is %d\n", getpid());*

*printf("The child’s parent process ID is %d\n", getppid());*

*}*

*else*

*{*

*/\* this is the parent process \*/*

*printf("The parent process ID is %d\n", getpid());*

*printf("The parent’s parent process ID is %d\n", getppid());*

*}*

*sleep(2);*

*}*

**Write a program to find the Turnaround time, waiting time using First come first serve scheduling algorithm**.

*#include<stdio.h>*

*int main()*

*{*

*int bt[20], wt[20], tat[20], i, n;*

*float wtavg, tatavg;*

*printf("\nEnter the number of processes -- ");*

*scanf("%d", &n);*

*for(i=0;i<n;i++)*

*{*

*printf("\nEnter Burst Time for Process %d -- ", i);*

*scanf("%d", &bt[i]);*

*}*

*wt[0] = wtavg = 0;*

*tat[0] = tatavg = bt[0];*

*for(i=1;i<n;i++)*

*{*

*wt[i] = wt[i-1] +bt[i-1];*

*tat[i] = tat[i-1] +bt[i];*

*wtavg = wtavg + wt[i];*

*tatavg = tatavg + tat[i];*

*}*

*printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");*

*for(i=0;i<n;i++)*

*printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);*

*printf("\nAverage Waiting Time -- %f", wtavg/n);*

*printf("\nAverage Turnaround Time -- %f", tatavg/n);*

*return 0;*

*}*

|  |
| --- |
| **related questions:**   1. **Write a program to implement some system call fork(), getpid(), getppid(), exit() and analysis the result obtained .** 2. **Write a program to implement FCFS with Turnaround time and waiting time and analysis the result obtained.** |