OPERATING SYSTEMS LAB-4

1. #include <stdio.h> #include <unistd.h> int main() {

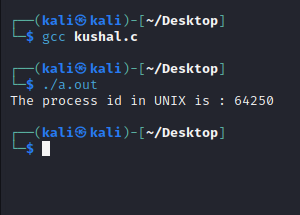
int p\_id;

p\_id = getpid();

printf("The process id in UNIX is : %d\n",p\_id);

return 0;

}



1. #include <stdio.h> #include <unistd.h> int main() {

int p\_id;

p\_id = getpid();

printf("The process id in UNIX is : %d\n",p\_id); fork();

printf(“Hello World\n”); return 0;

}



3)

#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

int pid; int i;

pid=getpid();

printf("The Process ID of Parent is : %d\n",pid); for(i=0;i<3;i++){

printf("Level %d",i); fork();

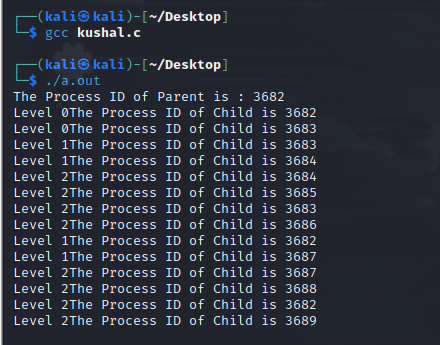
int pid=getpid();

printf("The Process ID of Child is %d\n",pid); wait(NULL);

}

return 0;

}



4)

#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

int pid; int i;

pid=getpid();

printf("The Process ID in UNIX is : %d\n",pid); if(fork() || fork()){

{

fork();

}

printf("Hello\n");}

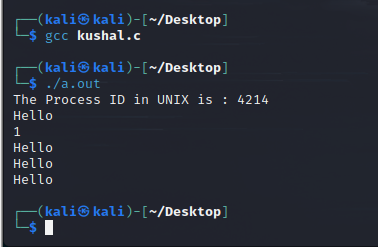
else{

printf("1\n");}

wait(NULL);

return 0;

}



5) #include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

int pid; int i;

pid=getpid();

printf("The Process ID in UNIX is : %d\n",pid); if(fork() && fork()){

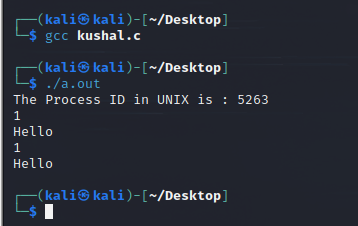
{

fork();

}

printf("Hello\n");} else{ printf("1\n");}

wait(NULL); return 0;



6)

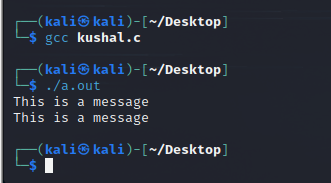
#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

fork();

printf("This is a message\n"); return 0;

}



7) #include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

pid\_t pid; pid=fork(); if (pid==0){

{

printf("Child Process\n");

}}

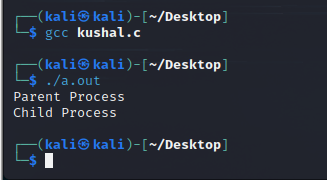
else

{

printf("Parent Process\n");}

return 0;

}



8)

#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

pid\_t pid; pid=fork(); int x=1;

if (pid==0){

{

printf("Child Process equals to : %d\n",x+1);

}}

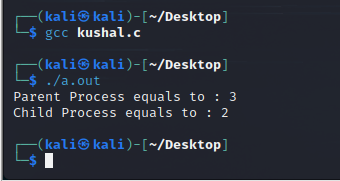
else

{

printf("Parent Process equals to : %d\n",x+2);}

return 0;

}



9)

include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h>

int main(){

pid\_t pid; pid=getpid(); int x=1; fork();

if (pid==0){

{

printf("Child Process equals to : %d\n and PPID is equal to

%d\n",x+1,getpid());

}}

else

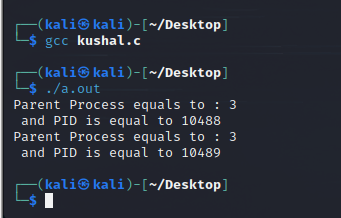
{

printf("Parent Process equals to : %d\n and PID is equal to

%d\n",x+2,getpid());}

return 0;

}



10)

#include<stdio.h>

#include<sys/types.h> #include<stdlib.h> #include<unistd.h>

int main(){

pid\_t pid; pid=fork();

if (pid==0) //Child Process

{

printf("Child process is executing\n");

}

if(pid>0)

{

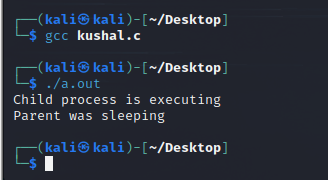
sleep(10);

printf("Parent was sleeping\n"); exit(0); //Parent Process

}

return 0;

}



Parent sleeps for 10s and then prints output.

11)

#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<stdlib.h> #include<unistd.h>

int main(){

pid\_t pid; pid=fork();

if (pid==0) //Child Process

{

sleep(10); printf("Child\n");

}

if(pid>0)

{

printf(“Parent\n”);

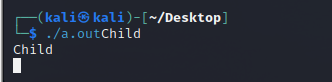
wait(NULL);

printf("I am in parent process.\n"); exit(0); //Parent Process

}

return 0;

}



It will print parent first and then after sleeping for 10s it will print Child and after the wait status changes , it will print I am in parent process.

12)

#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <sys/types.h> #include <sys/wait.h> int main(void)

{

pid\_t pid = fork();

if ( pid == 0 )

{

printf(“Child process is executing.\n”);

}

int status;

waitpid(pid, &status, 0); if ( WIFEXITED(status) )

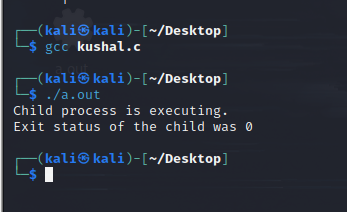
{

int exit\_status = WEXITSTATUS(status);

printf("Exit status of the child was %d\n",exit\_status);

}

return 0;}



13)

#include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/wait.h> #include <unistd.h>

int main(void) { pid\_t pid = fork();

if(pid == 0) {

printf("Child => PPID: %d PID: %d\n", getppid(), getpid()); exit(0);

}

else if(pid > 0) {

printf("Parent => PID: %d\n", getpid()); printf("Waiting for child process to finish.\n"); wait(NULL);

printf("Child process finished.\n");

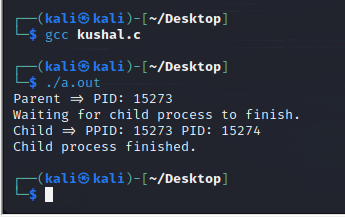
}

else {

printf("Unable to create child process.\n");

}

return 0;}



14)

#include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include <sys/wait.h> #include <unistd.h>

int main(void) { pid\_t pid = fork();

if(pid == 0) {

execl("/bin/ls", "ls", "-l", "/home/nitink/LAB-4",NULL); exit(0);

}

else if(pid > 0) {

printf("Parent => PID: %d\n", getpid()); printf("Waiting for child process to finish.\n"); wait(NULL);

printf("Child process finished.\n");

}

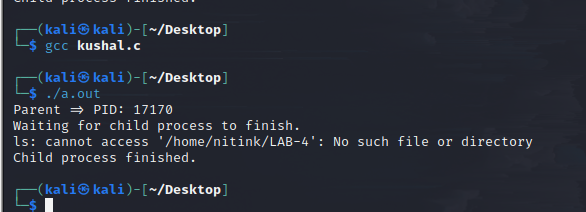
else {

printf("Unable to create child process.\n");

}

return 0;

}



EXTRA:

1)

#include <stdio.h> #include <unistd.h>

int main()

{

int pid, i;

printf("Start of main...\n");

pid = fork();

if (pid > 0) { printf("Parent section

\n");

}

else if (pid == 0) { printf("\nFork created\n");

}

else {

printf("\nFork creation failed\n");

}

printf("Printing the numbers from 1 to 10\n"); for (i = 1; i <= 10; i++)

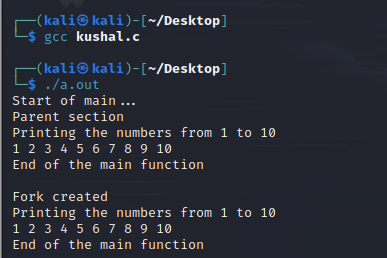
printf("%d ", i);

printf("\n");

printf("End of the main function\n");

return 0;

}



2)

#include <iostream> #include <unistd.h> int main()

{

int OddSum = 0, EvenSum = 0; int pid, i;

pid = fork();

if (pid > 0) {

for (i = 0; i < 10; i++) { if (a[i] % 2 == 0)

EvenSum = EvenSum + a[i];

}

printf("Parent process:\n");

printf("Sum of even no. is %d\n",EvenSum);

}

else {

for (i = 0; i < 10; i++) { if (a[i] % 2 != 0)

OddSum = OddSum + a[i];

}

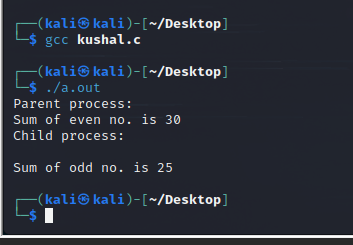
printf("Child process:\n");

printf("\nSum of odd no. is %d\n",OddSum);

}

return 0;

}



3)

#include<stdio.h> #include<sys/types.h> #include<sys/wait.h> #include<unistd.h> #include<stdlib.h>

int main()

{

int n;

printf("Enter the number of process you want to print: "); scanf("%d",&n);

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

{

if(fork() == 0)

{

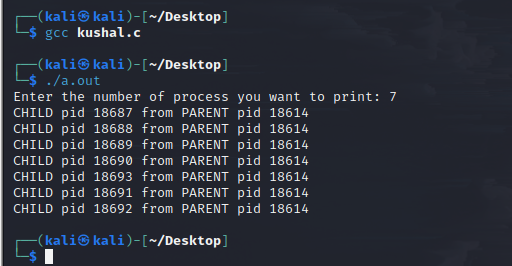
printf("CHILD pid %d from PARENT pid %d\n",getpid(),getppid()); exit(0);

}

}

for(int i=0;i<n;i++) wait(NULL); return 0;

}



Done by

Gundala Kushal bhavani reddy

CB.EN.U4CYS21021