#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

int main()

{

int status;

pid\_t pid;

pid = fork();

if(pid==0)

{

printf("\nI am Child , My pid is %d and My Parent id is %d",getpid(),getppid());

exit(0);

}

else

{

wait(&status);

printf("\nI am Parent , My id is %d and My Child id is %d\n",getpid(),pid);

exit(0);

}

}

#include<stdio.h>

#include<stdlib.h>

#include<sys/wait.h>

#include<string.h>

int main()

{

printf("\n$ ps\n");

char command[3];

strcpy(command,"ps");

system(command);

return 0;

}

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

printf("\n 2 + 4 = ");

printf("\t(Wait for 5 seconds before giving output)\t");

sleep(5);

printf("6\n");

return 0;

}

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

int main()

{

int status;

pid\_t pid;

if(fork()==0)

exit(0);

else

pid = sleep(2);

printf("\nParent id is: %d",getpid());

printf("\n Child id is: %d\n",pid);

return 0;

}

#include<stdio.h>

#include<signal.h>

#include<sys/wait.h>

#include<stdlib.h>

void handle\_sigint(int sig)

{

printf("Caught Signal %d\n", sig);

kill(getpid(),sig);

printf(" End");

exit(0);

}

int main()

{

pid\_t pid;

int status;

if((pid=fork())==0)

{

printf("\nChild Process ID : %d" ,pid );

}

else

{

wait(&status);

printf("\nParent Process ID : %d ",getpid());

signal(SIGINT, handle\_sigint);

while(1);

}

printf(" End");

return 0;

}

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

if(fork()==0)

printf("\nChild process");

else

{

printf("\nParent process");

wait(NULL);

printf("\nChild process terminated");

}

printf("\nEnd\n");

return 0;

}

a)

STEP 1: DECLARE VARIABLE INTEGER PID,STATUS.

STEP 2: PID := fork().

STEP 3: IF PID: =0 THEN

                     PRINT: I AM CHILD

                     EXIT.

             ELSE

                     wait(&STATUS)

                     PRINT: I AM PARENT

                     EXIT.

b)

STEP 1: DECLARE VARIABLE CHARACTER STRING COMMAND.

STEP 2: COPY STRING "ps" TO COMMAND.

STEP 3: system(COMMAND).

STEP 4: EXIT.

c)

STEP 1: PRINT: 2+4 = .

STEP 2: sleep(5) [PROCESS SLEEP FOR 5 SECOND].

STEP 3: PRINT: 6

STEP 4: EXIT.

d)

STEP 1:  DECLARE VARIABLE INTEGER PID.

STEP 2: PID := fork().

STEP 3: IF PID: =0 THEN

                     EXIT.

             ELSE

                     PID := sleep(2);

STEP 4: PRINT: PARENT ID IS getpid() AND CHILD ID IS PID.

STEP 5: EXIT.

e)

SIGNAL(SIG)

STEP 1: kill(getpid(),SIG).

STEP 2: EXIT.

ALGORITHM SIGNALHANDLE()

STEP 1:  DECLARE VARIABLE INTEGER PID,STATUS.

STEP 2: IF (PID := fork()) :=0 THEN

                     PRINT: CHILD ID ID PID.

             ELSE

                     wait(STATUS)

                     PRINT: PARENT ID IS getpid()

                     SIGNAL(SIGINT,handle\_sigint).

                     WHILE 1 THEN.

STEP 3: EXIT.

f)

STEP 1: IF fork() :=0 THEN

                     PRINT: I AM CHILD.

             ELSE

                     wait(NULL)

                     PRINT: I AM PARENT.

STEP 2: EXIT.

student@4bc6e9e7f9c9:~/Elab$ gcc lab1a.c

student@4bc6e9e7f9c9:~/Elab$ ./a.out

I am Child , My pid is 51 and My Parent id is 50

I am Parent , My id is 50 and My Child id is 51

student@4bc6e9e7f9c9:~/Elab$ gcc lab1b.c

student@4bc6e9e7f9c9:~/Elab$ ./a.out

$ ps

PID TTY TIME CMD

33 pts/2 00:00:00 bash

57 pts/2 00:00:00 a.out

58 pts/2 00:00:00 sh

59 pts/2 00:00:00 ps

student@4bc6e9e7f9c9:~/Elab$ gcc lab1c.c

student@4bc6e9e7f9c9:~/Elab$ ./a.out

2 + 4 = (Wait for 5 seconds before giving output) 6

student@4bc6e9e7f9c9:~/Elab$ gcc lab1d.c

student@4bc6e9e7f9c9:~/Elab$ ./a.out

Parent id is: 71

Child id is: 0

student@4bc6e9e7f9c9:~/Elab$ gcc lab1e.c

lab1e.c: In function 'handle\_sigint':

lab1e.c:9:10: warning: implicit declaration of function 'getpid'; did you mean 'getenv'? [-Wimplicit-function-declaration]

kill(getpid(),sig);

^~~~~~

getenv

lab1e.c: In function 'main':

lab1e.c:17:13: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]

if((pid=fork())==0)

^~~~

student@4bc6e9e7f9c9:~/Elab$ ./a.out

Child Process ID : 0 End

^CParent Process ID : 78 Caught Signal 2

Endstudent@4bc6e9e7f9c9:~/Elab$ gcc lab1f.c

student@4bc6e9e7f9c9:~/Elab$ ./a.out

Child process

End

Parent process

Child process terminated

End

student@4bc6e9e7f9c9:~/Elab$