Q1.

Program:-

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int pid = fork();

if (pid > 0)

{

printf("Parent process\n");

printf("id = %d\n", getpid());

}

else if (pid == 0)

{

printf("Child process\n");

printf("id = %d\n",getpid());

printf("Parent id : %d\n", getpid());

sleep(10);

printf("Child process \n");

printf("id=%d\n", getpid());

printf("Parent id = %d\n", getpid());

}

else

{

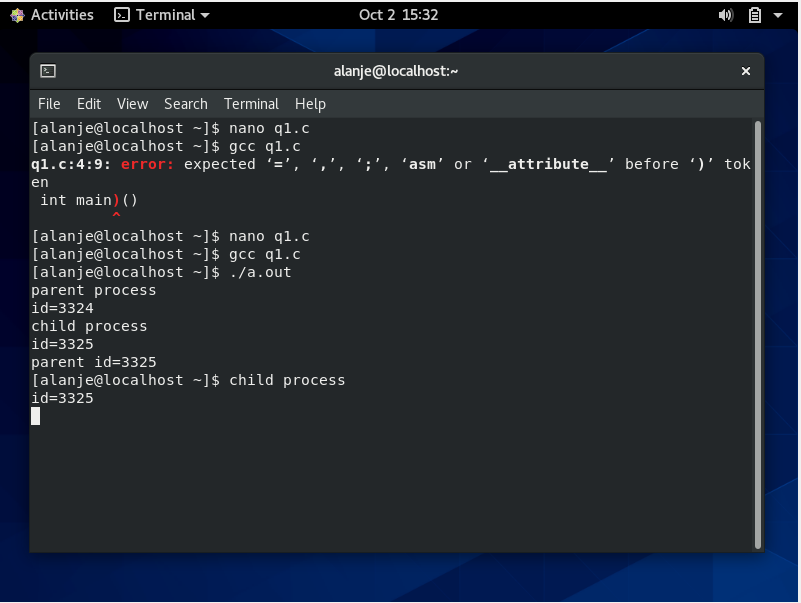
printf("Failed to create CP");

}

return 0;

}

Output:-



Q2.

Program:-

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include<stdio.h>

#include <sys/wait.h>

Int main()

{

Pid\_t id;

Id = fork();

If(id>0)

{

Printf(“parent going to sleep\n”);

Sleep(20);

Wait(NULL);

}

else

{

Printf(“child exited\n”);

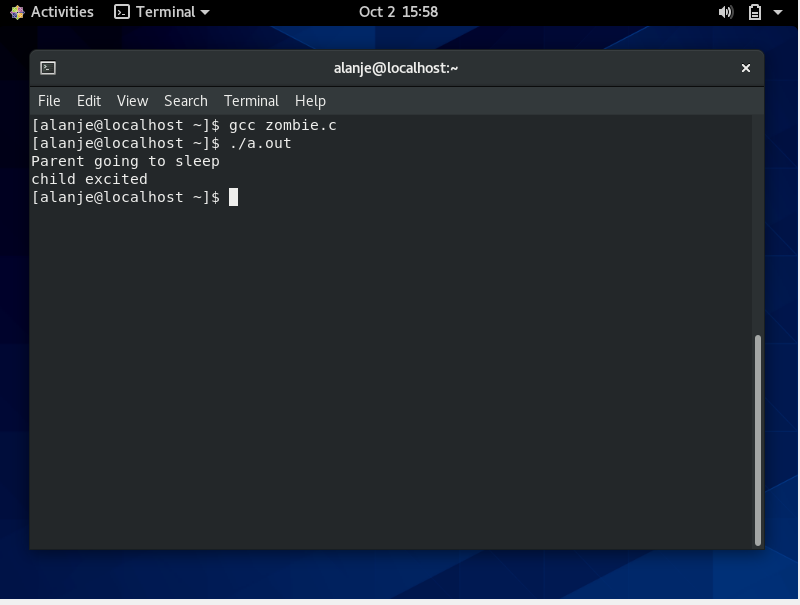
exit(0);

}

return 0;

}

Output:-



Q3.

Program:-

#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)

{

execl("/bin/ashish", "bin/ashish","-c","./nopath", "NULL");

}

int status;

waitpid(pid, &status, 0);

if (WIFEXITED(status))

{

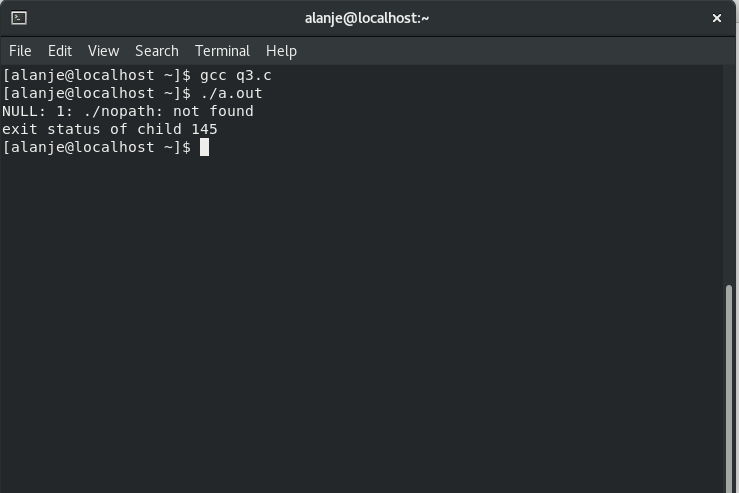
int exit\_status = WEXITSTATUS(status);

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

}

return 0;

}

Output:- 

Q4.

Program:-

#include <stdio.h>

#include <unistd.h>

int main()

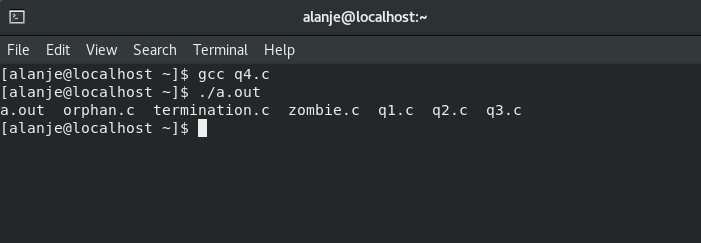
{

execl("/bin/ashish","ashish", NULL);

printf("rep\n");

}

Output:-



Q5.

Program:-

#include<pthread.h>

#include<stdio.h>

#include<stdlib.h>

void\* thread1(void\* arg1)

{

printf("thread1 executed\nStarting Firefox\n");

system("firefox");

}

void\* thread2(void\* arg2)

{

printf("thread2 executed\nStart Atom\n");

system("zoom");

pthread\_exit(0);

}

int main()

{

pthread\_t tid1,tid2;

pthread\_create(&tid1,NULL,thread2,(void\*)&tid1);

pthread\_create(&tid2,NULL,thread1,(void\*)&tid2);

pthread\_join(tid1,NULL);

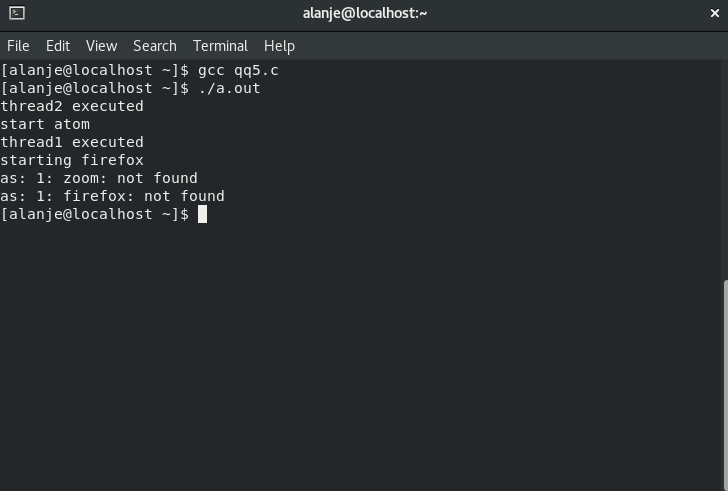
pthread\_join(tid2,NULL);

exit(0);

return 0;

}

Output:-



Q6.

Program:-

#include <signal.h>

#include<stdio.h>

#include <stdlib.h>

#include<sys/types.h>

#include<unistd.h>

void sighup();

void sigint();

void sigquit();

void main()

{

int pid;

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

{

perror("fork");

exit(1);

}

if (pid ==0)

{

signal(SIGHUP, sighup);

signal(SIGINT, sigint);

signal(SIGQUIT, sigquit);

for(;;);

}

else

{

printf("\nparents are sending signup\n\n");

kill(pid, SIGHUP);

sleep(3);

printf("\n parents are sending sigint\n\n ");

kill(pid, SIGINT);

sleep(3);

printf("\nparents are sending sigquit\n\n");

kill(pid, SIGQUIT);

sleep(3);

}

}

void sighup()

{

signal(SIGHUP, sighup);

printf("child received sighup\n");

}

void sigint()

{

signal(SIGINT, sigint);

printf("child received sigint\n");

}

void sigquit()

{

printf("parents blocked child\n");

exit (0);

}

Output:-

