Write a C/C++ program to illustrate the race condition.

#include<stdio.h>

#include<sys/types.h>

void charatatime(char \*str)

{

char \*ptr;

int c;

setbuf(stdout,NULL);

for(ptr=str;((c=\*ptr++)!='\0');)

putc(c,stdout);

}

int main()

{

pid\_t pid=fork();

if(pid==0)

{

charatatime("child is in execution....");

}

else

{

charatatime("parent is in execution....");

}

return 0;

}

Write a C/C++ program that creates a zombie and then calls system to execute the ps command to verify that the process is zombie.

#include<stdio.h>

#include<stdlib.h>

int main()

{

pid\_t pid;

pid=fork();

if(pid<0)

{

printf("fork failure");

exit(0);

}

else if(pid==0)

{

printf("child pid %d\n",getpid());

exit(0);

}

else

{

sleep(3);

printf("Parent process\n");\

system("ps -l");

}

return 0;

}

Write a C/C++ program to avoid zombie process by forking twice.

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

int main()

{

pid\_t pid;

pid=fork();

if(pid==0)

{

pid=fork();

if(pid==0)

{

sleep(10);

printf("Second child parent process id is %d\n",getppid());

exit(0);

}

else

exit(0);

}

else

{

if(waitpid(pid,NULL,0)!=pid)

printf("waitpid error");

exit(0);

}

return 0;

}