Code1:

// code to create three child

// process of a parent

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

// Driver code

int main()

{

int pid, pid1, pid2;

// variable pid will store the

// value returned from fork() system call

pid = fork();

// If fork() returns zero then it

// means it is child process.

if (pid == 0) {

// First child needs to be printed

// later hence this process is made

// to sleep for 3 seconds.

sleep(3);

// This is first child process

// getpid() gives the process

// id and getppid() gives the

// parent id of that process.

printf("child[1] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid1 = fork();

if (pid1 == 0) {

sleep(2);

printf("child[2] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

else {

pid2 = fork();

if (pid2 == 0) {

// This is third child which is

// needed to be printed first.

printf("child[3] --> pid = %d and ppid = %d\n",

getpid(), getppid());

}

// If value returned from fork()

// in not zero and >0 that means

// this is parent process.

else {

// This is asked to be printed at last

// hence made to sleep for 3 seconds.

sleep(3);

printf("parent --> pid = %d\n", getpid());

}

}

}

return 0;

}

Code 26:

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<string.h>

#include<stdlib.h>

int main( int argc, char\* argv[] ) {

int fdone[2];

pid\_t childid;

char readBuff[50];

char writeBuff[50];

int readCounter;

pipe( fdone );

if( argc < 3 ) {

printf( "Atleast need 2 params " );

exit(1);

}

int fileOpen = open( argv[1], 0 );

int targetFile = open( argv[2], 0666 );

if ( fileOpen == -1 || targetFile == -1 ) {

printf( "Opening file failed " );

exit(1);

}

childid = fork();

if( childid == 0 ) {

// inside the child prcocess

close( fdone[1] );

read( fdone[0], readBuff, sizeof( readBuff ) );

printf( "The recived string is : %s", readBuff );

//Writing to the target fileOpen

write( targetFile, readBuff, strlen( readBuff ) + 1 );

} else {

// inside the parent process

close( fdone[0] );

// code to read from a text file

while( (readCounter = read( fileOpen, readBuff, sizeof( readBuff ) ) > 0 ) ) {

write( fdone[1], readBuff, sizeof( readBuff ) );

}

close( fdone[1] );

}

}