**#include** <unistd.h>

**#include** <stdio.h>

**int** **main**()

{

    pid\_t t1, t2;

    t1 = **fork**();

**if** (t1 **<** 0)

    {

**printf**("fork error t1=%d\n", t1);

**return** 0;

    }

**else** **if** (t1 **==** 0)

    {

**printf**("C");

    }

**else**

    {

        t2 = **fork**();

**if** (t2 **<** 0)

        {

**printf**("fork error t2=%d\n", t2);

**return** 1;

        }

**else** **if** (t2 **==** 0)

        {

**printf**("B");

        }

**else**

        {

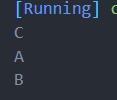
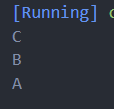
**printf**("A\n");

        }

    }

**return** 0;

}



**#include** <unistd.h>

**#include** <stdio.h>

**int** **main**()

{

**int** m = 10;

**int** i = 0;

    pid\_t t1, t2;

    t1 = **fork**();

**if** (t1 **<** 0)

    {

**printf**("fork error t1=%d\n", t1);

**return** 0;

    }

**else** **if** (t1 **==** 0)

    {

**for** (**int** i = 0; i **<** m; i++)

        {

**printf**("C");

        }

**printf**("\n");

    }

**else**

    {

        t2 = **fork**();

**if** (t2 **<** 0)

        {

**printf**("fork error t2=%d\n", t2);

**return** 1;

        }

**else** **if** (t2 **==** 0)

        {

**for** (**int** i = 0; i **<** m; i++)

            {

**printf**("B");

            }

**printf**("\n");

        }

**else**

        {

**for** (**int** i = 0; i **<** m; i++)

            {

**printf**("A");

            }

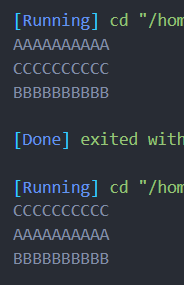
**printf**("\n");

        }

    }

**return** 0;

}



//1000太大 控制台装不下,所以改成打印十个

**#include** "stdio.h"

**#include** <string.h>

**#include** <sys/wait.h>

**#include** <unistd.h>

**int** **main**()

{

**int** pipe\_fd[2];

**int** status = **pipe**(pipe\_fd);

**if** (status **==** -1)

    {

**printf**("pipe create failed\n");

**return** -1;

    }

    \_\_pid\_t pid1, pid2;

    pid1 = **fork**();

**if** (pid1 **<** 0)

    {

**printf**("function fork() error\n");

**printf**("pid1 = %d \n ", pid1);

    }

**else** **if** (pid1 **==** 0)

    {

        //*printf("pid1 = %d \n pid2 = %d\n",pid1, pid2);*

**char** buf[1000] = "Child 1 is sending a message!\n";

**if** (**write**(pipe\_fd[1], buf, **strlen**(buf)) **!=** -1)

        {

**printf**("child pid = %d parent pid = %d child 1 send success\n", **getpid**(), **getppid**());

        }

**else**

        {

**printf**("child1 send error\n");

        }

    }

**else**

    {

        pid2 = **fork**();

**if** (pid2 **<** 0)

        {

**printf**("function fork() error\n");

**printf**("pid2 = %d \n ", pid2);

**return** 1;

        }

**else** **if** (pid2 **==** 0)

        {

            //*printf("pid1 = %d \n pid2 = %d\n",pid1, pid2);*

**char** buf[1000] = "Child 2 is sending a message!\n";

**if** (**write**(pipe\_fd[1], buf, **strlen**(buf)) **!=** -1)

            {

**printf**("child pid = %d parent pid = %d child2 send success\n", **getpid**(), **getppid**());

            }

**else**

            {

**printf**("child2 send error\n");

            }

        }

**else**

        {

            //*printf("parent pid = %d\n",getpid());*

**printf**("parent gets the mes!\n");

**int** rnum = 0, i = 0;

**char** buf[100];

**while** (**wait**(**NULL**) **!=** -1)

            {}

            //*wait(pid1, NULL, 0);*

            //*wait(pid2, NULL, 0);*

**close**(pipe\_fd[1]);

            rnum = **read**(pipe\_fd[0], buf, 100);

**if** (rnum **>** 0)

            {

**for** (i = 0; i **<** rnum; i++)

                {

**printf**("%c", buf[i]);

                }

            }

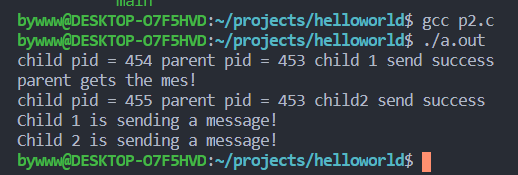
**close**(pipe\_fd[0]);

        }

    }

**return** 0;

}



//弃用的代码

**#include** "stdio.h"

**#include** <string.h>

**#include** <unistd.h>

**int** **main**(){

**int** pipe\_fd[2];

**int** status = **pipe**(pipe\_fd);

**if**(status **==** -1){

**printf**("pipe create failed\n");

**return** -1;

    }

    \_\_pid\_t pid1,pid2;

    pid1 = **fork**();

    pid2 = **fork**();

**if**(pid1**<**0 **||** pid2 **<**0){

**printf**("function fork() error\n");

**printf**("pid1 = %d \n pid2 = %d\n",pid1, pid2);

    }

**else** **if**(pid1 **==**0 **&&** pid2 **==**0){

        //

    }

**else** **if**(pid1 **==**0 ) {

        //*printf("pid1 = %d \n pid2 = %d\n",pid1, pid2);*

**char** buf[1000] = "Child 1 is sending a message!\n";

**if**(**write**(pipe\_fd[1],buf,**strlen**(buf)) **!=** -1){

**printf**("child pid = %d parent pid = %d child 1 send success\n",**getpid**(), **getppid**());

        }

**else**{

**printf**("child1 send error\n");

        }

    }**else** **if**(pid2 **==** 0) {

        //*printf("pid1 = %d \n pid2 = %d\n",pid1, pid2);*

**char** buf[1000] = "Child 2 is sending a message!\n";

**if**(**write**(pipe\_fd[1],buf,**strlen**(buf)) **!=** -1){

**printf**("child pid = %d parent pid = %d child2 send success\n",**getpid**(),**getppid**());

        }

**else**{

**printf**("child2 send error\n");

        }

    }**else** {

        //*printf("parent pid = %d\n",getpid());*

**printf**("parent gets the mes!\n");

**int** rnum = 0,i=0;

**char** buf[100];

**wait**(pid1,**NULL**,0);

**wait**(pid2,**NULL**,0);

**close**(pipe\_fd[1]);

        rnum = **read**(pipe\_fd[0],buf,100);

**if**( rnum **>** 0) {

**for**(i=0;i**<**rnum;i++){

**printf**("%c",buf[i]);

            }

        }

**close**(pipe\_fd[0]);

    }

**return** 0;

}

