Lab1.c

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

int pid;

pid=fork();

switch(pid)

{

case -1:

printf("fork fail!\n");

exit(1);

case 0: //子进程打开vi编辑器部分

execl("/usr/bin/vi","vi","/home/yxj/Desktop/OStest2",NULL);

printf("exec fail!\n");

exit(1);

default: //主进程执行for循环维持进程正常执行

for (int i=0;;i++){};

printf("vi completed!\n");

exit(1);

}

}

Lab2\_1.c kill停止

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

pid\_t p1, p2, p3, p4, p5;

while ((p1 = fork()) == -1);

if (!p1) {

while ((p2 = fork()) == -1);

if (!p2) {

while ((p4 = fork()) == -1);

if (!p4) {

while (1){

printf("Node p4 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getpid());

//getpid()输出当前进程id，getppid()输出当前进程的父进程的id。

}

}

else{

while ((p5 = fork()) == -1);

if (!p5) {

while (1){

printf("Node p5 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

}

while (1) {

printf("Node p2 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else {

while ((p3 = fork()) == -1);

if (!p3) {

while (1){

printf("Node p3 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

}

while (1) {

printf("Node p1 is a parent with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

return 0;

}

Lab2\_2.c正常退出

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

pid\_t p1, p2, p3, p4, p5;

while ((p1 = fork()) == -1);

if (!p1) {

while ((p2 = fork()) == -1);

if (!p2) {

while ((p4 = fork()) == -1);

if (!p4) {

while (1){

printf("Node p4 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getpid());

//getpid()输出当前进程id，getppid()输出当前进程的父进程的id。

}

}

else{

while ((p5 = fork()) == -1);

if (!p5) {

while (1){

printf("Node p5 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

}

while (1)

{

printf("Node p2 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

Exit(0);//正常退出

}

}

else {

while ((p3 = fork()) == -1);

if (!p3) {

while (1){

printf("Node p3 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

}

while (1) {

printf("Node p1 is a parent with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

return 0;

}

Lab2\_3.c 错误退出

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main() {

pid\_t p1, p2, p3, p4, p5;

while ((p1 = fork()) == -1);

if (!p1) {

while ((p2 = fork()) == -1);

if (!p2) {

while ((p4 = fork()) == -1);

if (!p4) {

while (1){

printf("Node p4 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getpid());

//getpid()输出当前进程id，getppid()输出当前进程的父进程的id。

}

}

else{

while ((p5 = fork()) == -1);

if (!p5) {

while (1){

printf("Node p5 is p2's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

}

while (1)

{

printf("Node p2 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

int \*p = NULL;

\*p = 0;//通过访问不存在的内存地址，引发段错误退出

}

}

else {

while ((p3 = fork()) == -1);

if (!p3) {

while (1){

printf("Node p3 is p1's child with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

}

while (1) {

printf("Node p1 is a parent with pid %d, it's parent pid %d.\n", getpid(), getppid());

}

}

else { ; }

return 0;

}