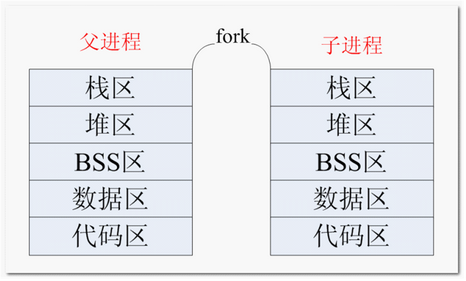
实验3 进程创建

1. 实验目的
2. 加深对进程概念的理解，明确进程与程序的区别；
3. 掌握创建进程的方法，进一步认识并发执行的本质。
4. 掌握Linux的库函数fork()，加深对进程的理解。
5. 理解Linux中线程的创建。
6. 预备知识
7. 头文件介绍

* pthread.h gcc下的线程库。
* sys/types.h 中文名称 为 基本系统数据类型，此[头文件](https://so.csdn.net/so/search?q=%E5%A4%B4%E6%96%87%E4%BB%B6&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_34669783/article/details/_blank)还包含适当时应使用的多个基本派生类型pid\_t 进程ID和进程组ID
* unistd.h所定义的接口通常都是大量针对系统调用的封装（英语：wrapper functions），如 fork、pipe 以及各种 I/O 原语（read、write、close 等等）。

1. 函数介绍

Linux中创建进程的常用方法是使用fork() 函数从已存在的进程（父进程）中创建一个新进程（子进程）。子进程是父进程的副本，子进程和父进程使用相同的代码段；子进程复制父进程的数据与堆栈空间，并继承父进程的用户代码、组代码、环境变量、已打开的文件代码、工作目录和资源限制等。由于子进程几乎完全复制了父进程，因此父子进程会运行同一个程序。



fork() 函数的调用格式为：

int fork();

* 返回值的意义：

**正确返回：**等于0，表示当前进程是子进程，从子进程返回进程ID值。

大于0，表示当前进程是父进程，从父进程返回新的子进程的进程 ID值。

**错误返回：**小于0，表示进程创建失败。

子进程虽然继承了父进程的一切数据，但子进程一旦开始运行，就会和父进程分开。子进程拥有自己的进程ID、资源以及计时器等，其与父进程之间不再共享任何数据。

1. 实验内容
2. **进程创建**

功能：用于从一个已存在的进程中创建一个新进程，新进程称为子进程，原进程称为父进程。

参数：无

返回值：成功：子进程中返回 0，父进程中返回子进程 ID。pid\_t，为无符号整型。

失败：返回 -1。

失败的两个主要原因是：

1. 当前的进程数已经达到了系统规定的上限，这时 errno 的值被设置为 EAGAIN。
2. 系统内存不足，这时 errno 的值被设置为 ENOMEM。

**例1：获取创建的子进程号**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

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

{

if (fork()==0)

printf("id ==== %d\n", getpid()); // 获取创建的子进程号

return 0;

}

运行结果：

id ==== 5471

**例2：利用fock()函数创建并显示父子进程**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

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

{

pid\_t pid;

int count;

pid = fork();

if( pid < 0 ){ // 没有创建成功

perror("fork");

}

if(0 == pid){ // 子进程

while(count!=8){

printf("I am son\n");

sleep(1);

count++;

}

}else if(pid > 0){ // 父进程

while(count!=8){

printf("I am father\n");

sleep(1);

count++;

}

}

return 0;

}

运行结果：

I am father

I am son

I am father

I am son

I am father

I am son

I am father

I am son

I am father

I am son

I am father

I am son

I am father

I am son

I am father

I am son

**例3：验证父子进程地址是独立的**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int a = 10; // 全局变量

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

{

int b = 20; //局部变量

pid\_t pid;

pid = fork();

if( pid < 0 ){ // 没有创建成功

perror("fork");

}

if(0 == pid){ // 子进程

a = 111;

b = 222; // 子进程修改其值

printf("son: a = %d, b = %d\n", a, b);

}else if(pid > 0){ // 父进程

sleep(1); // 保证子进程先运行

printf("father: a = %d, b = %d\n", a, b);

}

return 0;

}

运行结果如下：

son: a = 111, b = 222

father: a = 10, b = 20

**例4：一个进程包括代码、数据和分配给该进程的资源。fork（）函数通过系统调用创建一个新进程，操作系统同时要给新进程分配资源。下面是一个创建进程的例子：（注意冒号、引号等符号要用英文状态输入！）**

#include <sys/types.h>

#include <unistd.h>

#include<stdio.h>

int main()

{

pid\_t fpid; //fpid表示fork函数返回的值

int count=0;

fpid=fork();

if (fpid<0)

printf(“error in fork!\n”);

else if (fpid==0)

{ printf(“I’m the child process, my process ID is %d\n”,getpid());

count++;

}

else { printf(“I’m the parent process, my process ID is %d\n”,getpid());

count++;

}

printf(“Total result: %d\n”, count);

return 0;

}

结果：

I’m the parent process, my process ID is 5398

Total result: 1

I’m the child process, my process ID is 5399

Total result: 1

1. **线程的创建**

* 在 Linux 下：线程就是轻量级的进程，对于内核来说，线程就是轻质进程（内核只会用）。查看指定线程的 LWP (Light weight Process)号，线程号和线程 ID 是有区别的，线程号是给内核看的。

查看方式，找到程序的进程 ID ：ps -Lf pid

例如，Linux 下 查看火狐浏览器，发现其是由多线程构成：

ps ajx | grep "firefox"

ps -Lf 31102

* pthread\_create()：

作用： 创建子线程

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);

主线程退出，子线程一定会被强制结束退出。

参数：

thread： 传出参数，线程创建成功后，会设置一个合适值，通常比较大。

attr： 线程属性，默认使用 NULL

start\_routine： 子线程的处理函数（也被称为回调函数）

arg： 子线程的处理函数的参数

返回值：函数调用成功返回 0，失败返回错误号；

**例5：使用不带参数的，子线程的处理函数，验证子线程执行的代码，只有该回调函数里面的内容。**

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

void\* myfun(void\* arg);

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

{

pthread\_t pthread = 0; //创建一个子线程

int ret = pthread\_create(&pthread, NULL, myfun, NULL);

printf("parent thread id: %ld\n", pthread\_self（）);

sleep(2); //避免主线程运行后，就死亡了，而子线程没机会

for (int i = 0; i < 5; i++)

{ //验证子线程，并不会执行这里面的代码，只会执行回调函数 muyfun 里面的

printf("i = %d\n", i);

}

return 0;

}

void\* myfun(void\* arg)

{

printf("child thread id: %ld\n", pthread\_self（）);

return NULL;

}

运行的命令：gcc ex3-5.c -o ex3-5 -lpthread（由于pthread 库不是标准Linux库，运行的时候要写后缀-lpthread）

然后输入：./ex3-5 运行，

运行结果：parent thread d: 139828963149632

child thread id: 139828963145472

i = 0

i = 1

i = 2

i = 3

i = 4

1. 实验习题
2. 进程创建

编写一段程序，使用系统调用fork（）创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符或一句话，例如父进程显示“parent”，子进程分别显示字符“1”和字符“2”。每位同学可以自行设置输出语句，试观察记录屏幕上的显示结果，并分析原因。

1. 编写一个C 程序，使用系统调用fork()以创建一个子进程，并使用这个子进程调用exec 函数族以执行系统命令ls。

示例代码：

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

pid\_t pid;

pid = fork(); /\* 创建子进程 \*/

if (pid < 0) { /\* 创建失败 \*/

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* 子进程\*/

execlp(“/bin/ls”,“ls”,NULL); /\* 装载子进程映像 ls 命令\*/

}

else { /\* 父进程\*/

wait (NULL); /\* 父进程等待子进程运行完毕 \*/

printf ("Child Complete");

}

1. 实验报告

将调试运行成功的程序和写好的实验报告一起压缩打包，以实验X-学号-姓名.rar这样的形式命名，并上传提交。