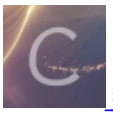
操作系统进程控制实验

原创

唐烫 于 2019-07-22 19:26:30 发布 2498 收藏 19

分类专栏： 操作系统 文章标签： 操作系统

本文链接：https://blog.csdn.net/qq_41785852/article/details/96888485



实验题目： 进程控制实验

实验学时：12 实验日期： 2019.5.2-2019.5.23

**

实验目的：

**

加深对于进程并发执行概念的理解。实践并发进程的创建和控制方法。观察和 体验进程的动态特性。进一步理解进程生命期期间创建、变换、撤销状态变换的过 程。掌握进程控制的方法，了解父子进程间的控制和协作关系。练习 Linux 系统中 进程创建与控制有关的系统调用的编程和调试技术。

-

硬件环境

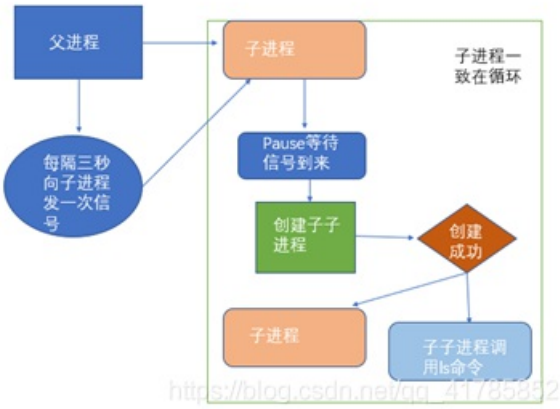
*： window

软件环境：

VWareUbuntu16.04

实验步骤与内容

1,首先仿照实验指导书给出的案例进行分析，发现并不能跑，真垃圾。里面的等于和赋值号不分，他的结果是P出来的吧。
2，独立实验要求：参考以上示例程序中建立并发进程的方法，编写一个父子协作进程，父进程创 建一个子进程并控制它每隔 3 秒显示一次当前目录中的文件名列表。
首先明确实验思路和流程



大体流程为：首先父进程创建子进程，父进程一直在执行循环体（每隔三秒向子进程发送一次信号）。对于子进程，如果创建成功，也一直在执行自己的循环体中（一直挂起等待信号唤醒，当被唤醒时，创建子子进程，子子进程用来执行调用函数（函数执行ls命令））。
实验代码如下：

```c
#include "mypctl.h"
int main(int argc, char *argv[])
 {
        int i;
        int pid;
        int status;
        int childpid;
        char *args[] = {"/bin/ls","-a",NULL};
        signal(SIGUSR1,(sighandler_t)sigcat);
        pid=fork();
        if(pid<0)  //SHI
        {
        printf("Create Process fail!\n");
        exit(EXIT_FAILURE);
        }
        if(pid==0)
        {

                while(1)
                {
                pause();
                childpid = fork();
                if(childpid<0)
                        exit(EXIT_FAILURE);
                if(childpid==0)
                        {
                                printf("%d childpid will Running:\n",getpid
());
```

```c
                                status= execve(args[0],args,NULL);
                        }
                }

        }
        else //father process
        {
                printf("%\d I am parent process %d\n",getpid());
                while(1)
                        {
                        sleep(3);
                        kill(pid ,SIGUSR1);
                        printf("%d parent sent signal:\n",getpid());
                        }

        }
         return EXIT_SUCCESS;
}
```



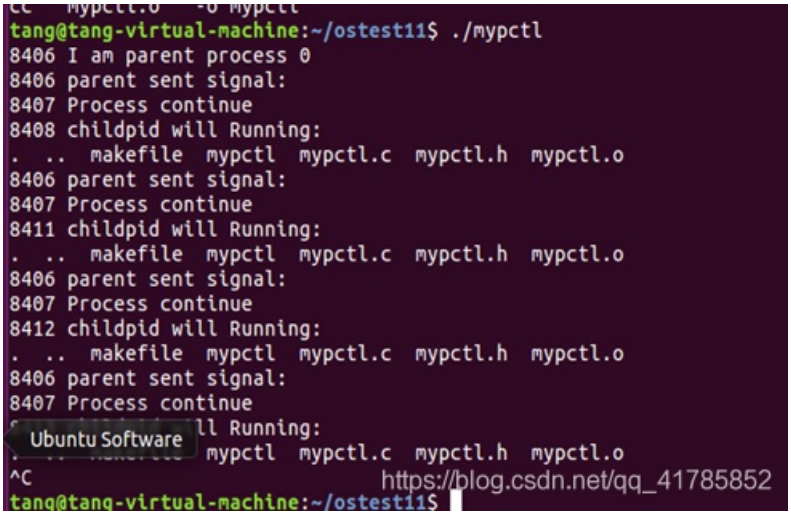
```c
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

typedef void(*sighandler_t)(int);
        void sigcat(){
                printf("%d Process continue\n",getpid());
        }
```

同时，创建啦一个makefile项目管理文件用来管理。

```
head = mypctl.h
srcs = mypctl.c
objs = mypctl.o
opts = -g -c
all:  mypctl
pctl:  $(objs)
        gcc $(objs) -o  mypctl
pctl.o: $(srcs)  $(head)
        gcc $(opts) $(srcs)
clean:
        rm mypctl  *.o
```

结论分析与体会

：
1） 结果：



运行结果如上图所示，首先执行父进程，每隔三秒父进程发送信号。
当打印process continue时得知子进程成功接收到信号。
当打印child will Running 时，可知子子进程创建成功，同时执行ls查看命令。

2） 分析

根据实验中观察和记录的信息结合示例实验和独立实验程序，说明它们反映出操 作系统教材中进程及处理机管理一节讲解的进程的哪些特征和功能？

1. 并行性，在执行父进程时，子进程也在执行
2. 动态性，通过实验可以清晰地观察进程地动态产生，动态消亡过程
3. 独立性，进程是调度的基本单位，也真是基于独立性，进程可以获得处理机并参与并发执行。
4. 异步性，每一个进程相对独立，比如一开始对pid的判断，无法预知是父进程先执行还是子进程先执行，是无法预知的，同时也体现啦进程是调度的基本单位。
   在真实的操作 系统中它是怎样实现和反映出教材中讲解的进程的生命期、进程的实体和进程状态 控制的。你对于进程概念和并发概念有哪些新的理解和认识？
5. 主要从子进程来看待一个进程的一生生命周期。子进程被创建（create）->进入调度序列就绪（ready）->执行（run）->进去就睡眠等待（体现为代码中的pause）->被唤醒->重新进入ready序列就绪->执行->终止。
6. 对进程的动态创建和消亡过程有啦更加直观的认识

子进程是如何创建和 执行新程序的？

1． 子进程是有父进程通过fork（）创建的，
2． 执行是通过调用execve（）函数，但值得注意的是当调用函数完成时，不能再回到原来的进程状态，相当于覆盖啦，所以在该实验中每一次要循环创建子子进程，让子子进程去执行函数调用，而子进程循环接收信号。

信号的机理是什么？

1.实际上就是一种软中断，在不同进程间传递信息。

怎样利用信号实现进程控制？
首先定义一个信号量，一个进程向另一个进程发送信号，接收信号的进程可以选择忽略信号，不做处理，可以类似中断处理程序，对于需要处理的信号，进程可以指定处理函数，进行处理。可以对信号的处理保留系统默认值，这种缺省操作。