实验2 2.1进程管理与进程创建

**实验目的**

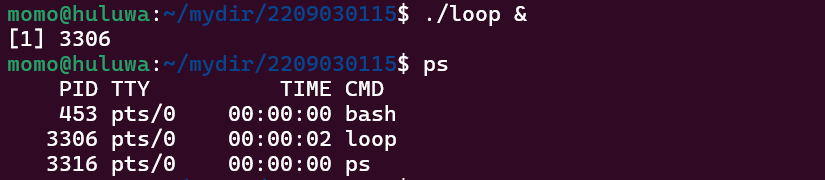
1. 掌握Linux中进程的创建方法及执行情况。
2. 加深对进程、进程树等概念的理解。

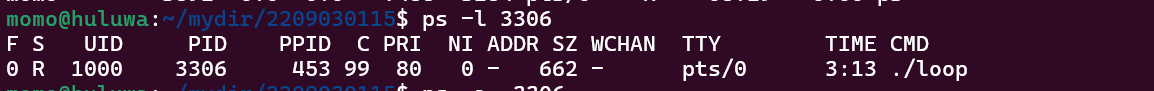
**实验内容**

**注意：所有操作和实验程序都要保存到自己学号命名的目录中！！**

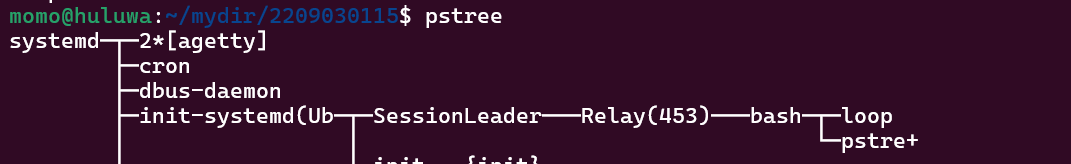
1. **进程操作命令（上传命令及运行截图）**

1.运行loop程序（实验1），使用ps命令查看该进程的详细信息并分析该进程的执行情况

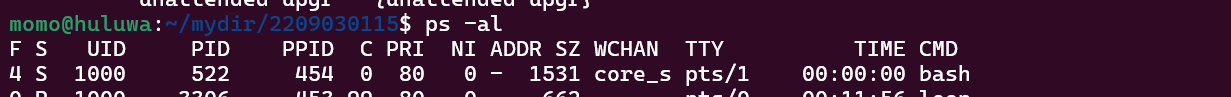




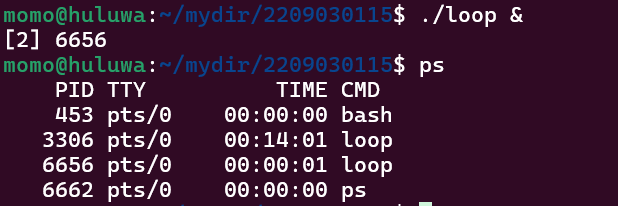
2.使用命令pstree查看loop进程所在的进程树，分析loop进程的父进程信息

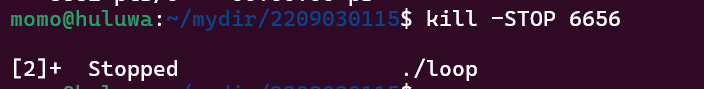


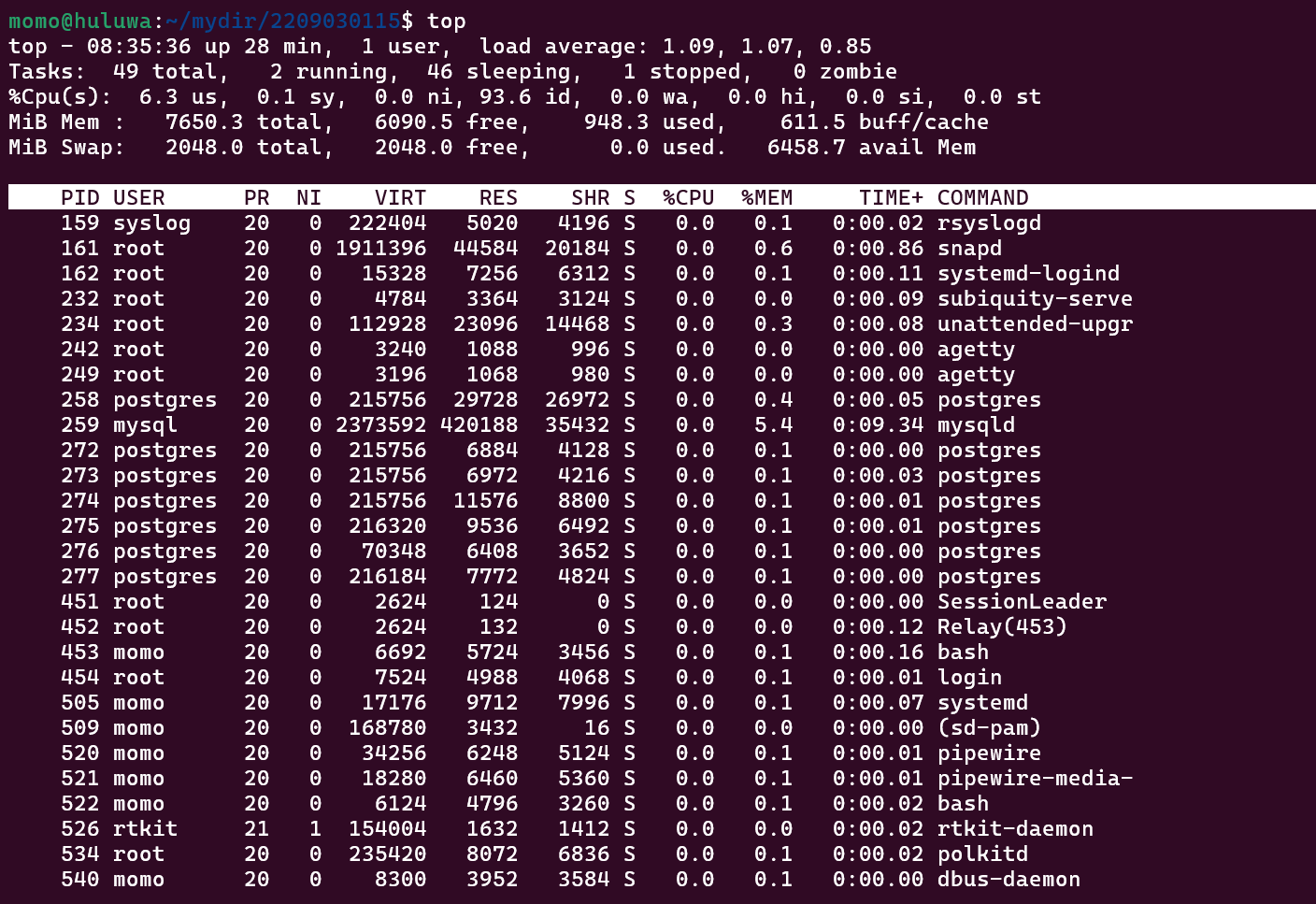
分析得父进程为bash，故父进程得信息为：



3.再运行一个loop程序，K并使用命令kill使该进程暂停，然后使用top命令查看自己Linux系统的进程情况，简要分析自己的系统的CPU和内存使用情况以及系统中的进程情况







通过图可知，用户空间占用cpu时间的百分比为6.3，内核空间占用cpu时间的百分比为0.1，空闲cpu的百分比为93.6.

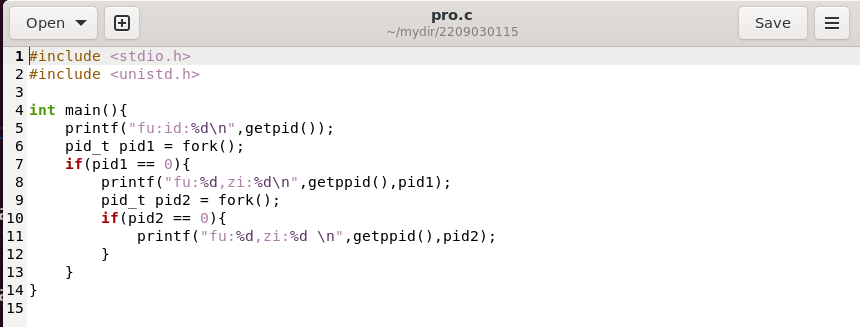
物理内存的总大小为7650.3，空闲内存为6090.5，已使用的内存为948.3，缓存为611.5.

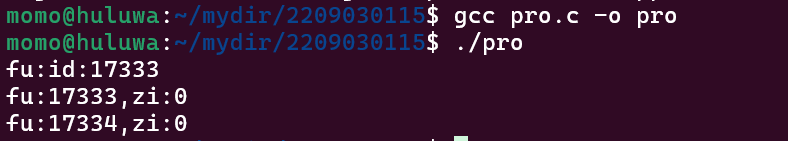
系统中共有49个任务，其中2个正在运行，46个在休眠，其中还有刚刚停止的loop进程。

1. **进程控制**

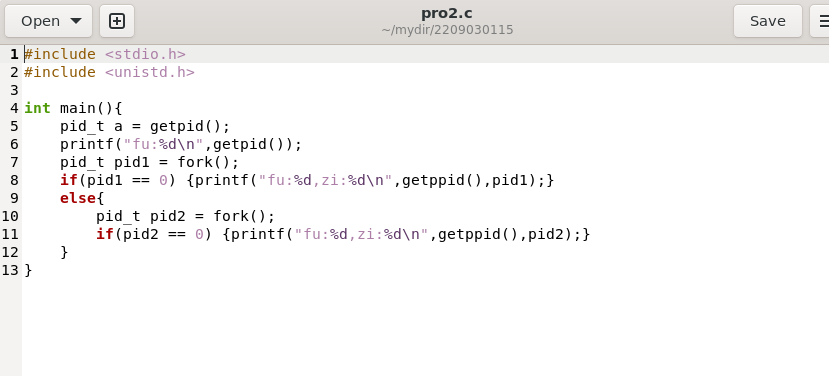
1.编写一段程序，使用系统调用fork( )创建两个子进程。

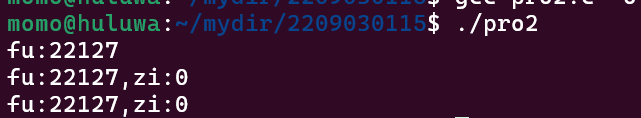
（1）父进程创建子进程1，子进程1创建子进程2，三个进程分别输出一行字符串，其中包含进程各自的pid和父进程id。





（2）父进程创建子进程1和子进程2，三个进程分别输出一行字符串，其中包含进程各自的pid和父进程id。





**要求：两种情况的源代码截图和运行结果截图。**

**注意分析运行结果是否符合程序设计。**

2.运行以下程序，分析程序执行过程中产生的进程情况。

/\*forktree.c\*/

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <string.h>

int main()

{

int p;

p=fork();

printf("fork 1\n");

if (p>0) /\*如果是父进程\*/

{

fork();

printf("fork 2\n"); /\*子进程和父进程重复执行\*/

}

else{ /\*如果是子进程\*/

fork();

printf("fork 3\n"); /\*子进程和孙进程重复执行\*/

fork(); /\*子进程和孙进程重复执行\*/

printf("fork 4\n"); /\*子进程、两个孙进程和曾孙进程重复执行\*/

}

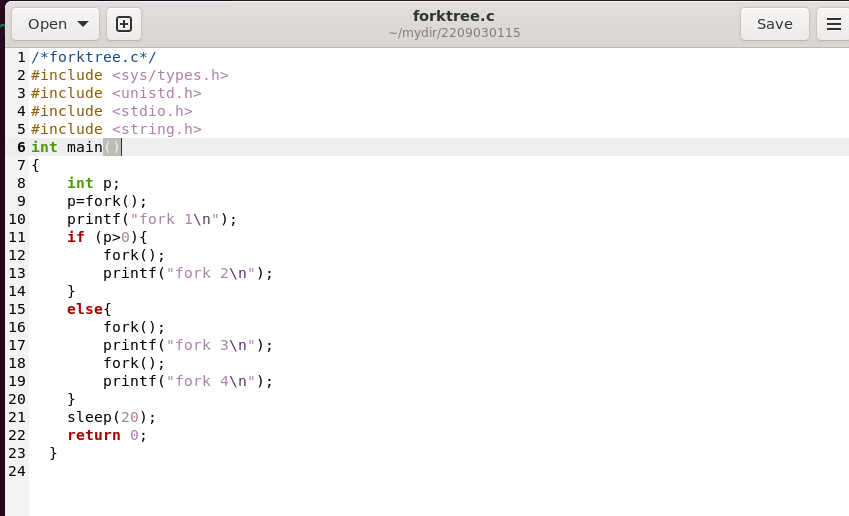
sleep(20);

return 0；

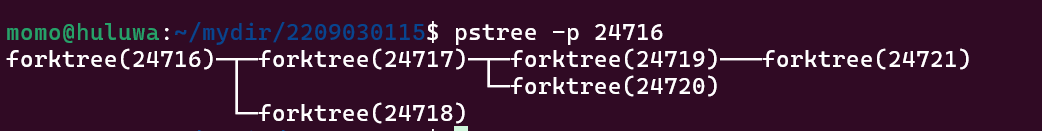
}

实验步骤：

* 编写上面的源代码生成源文件forktree.c



* 编译源文件生成可执行程序forktree：$gcc–o forktree forktree.c
* 后台运行：$./forktree &
* 查看进程树：$pstree –p pid （替换为forktree的进程id）或者用命令 ps axf查看几个进程的关联关系



* 分析：最后一个系统调用fork()会为什么被执行2次？最后一条printf输出语句为什么会被执行4次？

1.因为孙进程24719会进行一次fork语句生成曾孙进程24721，并且孙进程的父进程也会执行一次fork语句生成孙进程24719的兄弟进程24720.所以最后一个系统调用fork执行了两次。

2.最后一个曾孙进程执行一次printf fork 4，孙进程24719也会执行一次， 其兄弟进程24720会执行一次，并且其父进程也会执行一次。所以一共打印四次fork 4。

**要求：1.程序源代码和运行结果截图（注意截图要能看到forktree相关的进程树）**

**2.结果分析，并回答上述问题**

3.（选做）运行程序，分析以下程序运行产生的进程树情况。

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

int main()

{

int p,i;

for( i=0;i<2;i++)

{ p=fork();

if(p==0) exit(0);

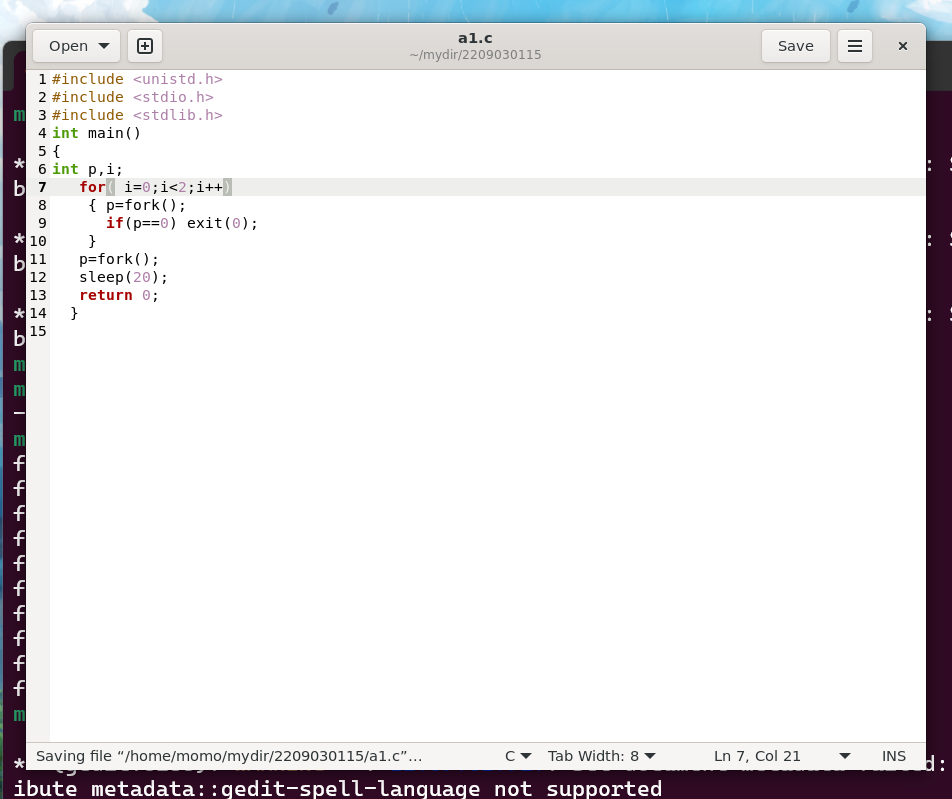
}

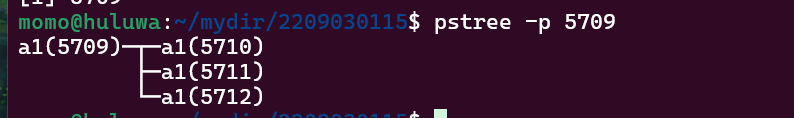
p=fork();

sleep(20);

return 0;

}





在父进程下进入循环，并执行fork()创建子进程，如果创建成功就退出该子进程并且重新在父进程下执行fork()，即创建了两次子进程。

退出循环后又在父进程下创建了一个新的子进程，所以一共创建了三个子进程。

4.（选做）分析验证课堂中的实例创建的进程情况

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

{

fork();

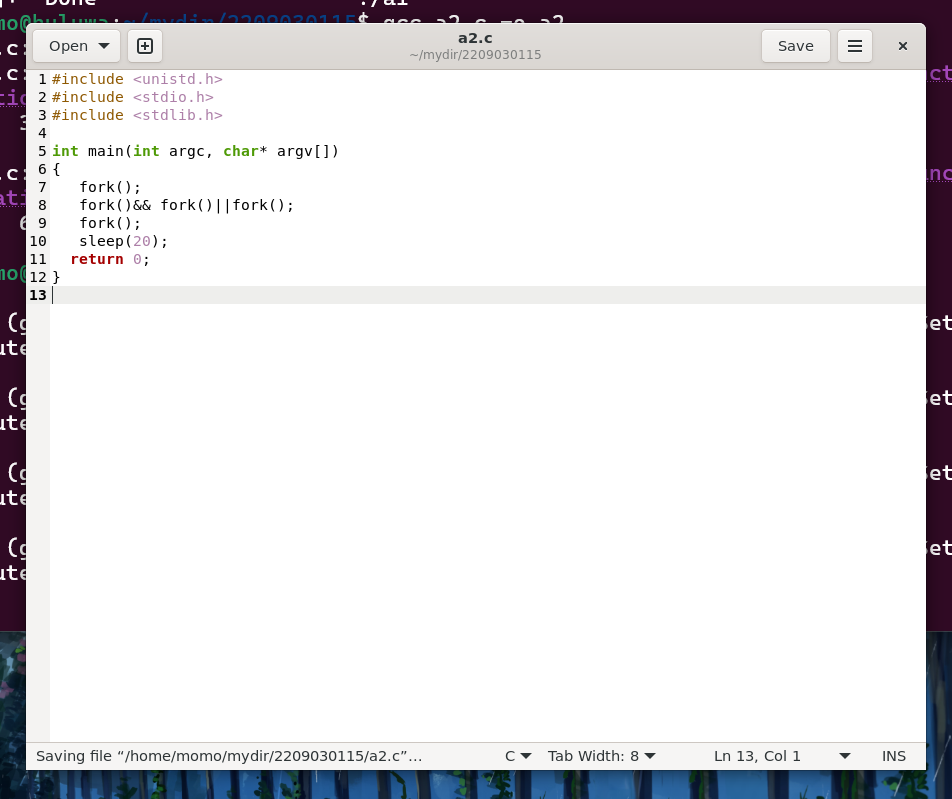
fork()&& fork()||fork();

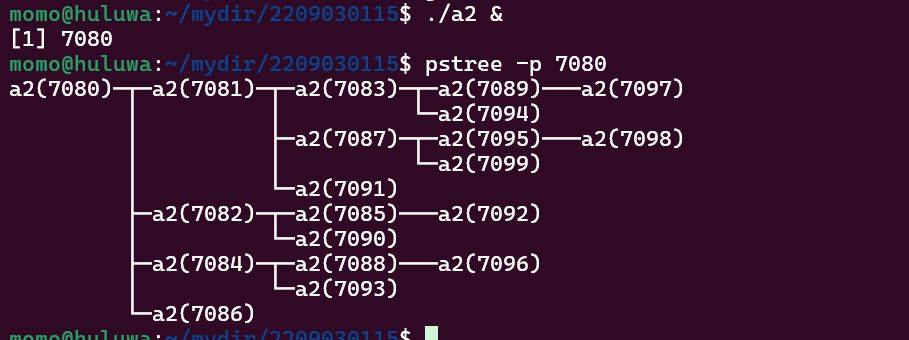
fork();

sleep(20);

return 0;

}





一共创建了十六个进程：

第一个fork产生两个进程父进程与子进程。

fork() && fork() || fork()中：

1. 第一个fork()：2个进程变为4个进程。
2. 第二个fork()（由于第一个fork()总是成功，所以第二个fork()在所有4个进程中执行）：4个进程变为8个进程。
3. 第三个fork()：由于第二个fork()总是成功，所以第三个fork()不会执行。

最后的fork()：8个进程变为16个进程。