

实验二实验报告

江学强, PB16120100

1. 阶段一：实现三个系统调用：fork、exec、join

添加系统调用的一般方法与实验一相同，不再赘述，只叙述实现的过程和遇到的一些问题，下面对三个系统调用分别叙述。

fork 系统调用：fork 系统调用的功能是创建一个新的子进程，子进程继承父进程的代码、内存空间和寄存器状态，fork 系统调用实现就是先要保存这些内容到子进程中，之后把子进程放到就绪队列，下面看 nachos 的实现。

实现 fork 系统调用的第一步就是新建一个 Thread 类作为子进程类，这里需要说明一下我是怎么实现每个进程唯一的 tid 的，我在 Thread 类中定义了一个静态的 unsigned int 变量 tidGenerate，初始化为 0，在构造函数中把 tidGenerate 赋值给 tid，并让 tidGenerate 递增，这样就实现了每个进程独特的 tid。此外我在构造函数中把 parent 变量赋值为 kernel->currentThread，即完成了父子进程关系的联系以及把新建的队列加入到 kernel 的进程队列中，调用 kernel->addThread。子进程新建后做一些保存的工作：保存内存空间，这里要注意保存之前要新建一个 AddrSpace 类实例给新建的对象，之后调用 AddrSpace 的类方法 CopyMemory，之后调用 SaveUserState 保存寄存器的值，这些工作做完之后调用 Thread 类方法 Fork，参数分别是一个函数指针，这个函数指针是处理子进程执行前的就绪过程的，就是 forked 函数，以及新建的 Thread 类。Fork 方法的用法我是在网上看了一篇调试 Nachos 的博客中学习了用 gdb 调试 Nachos 来单步跟踪提供的程序知道的用法，事实上这是 fork 系统调用实现的最难的一个点。

exec 系统调用：主要过程就是把执行的程序加载到内存并执行，下面看 nachos 的实现。

exec 的实现比较直接，情况内存空间调用 AddrSpace->reset()方法即可，之后再根据文件名把程序加载到内存，调用 AddrSpace->Load(filename)即可，如果加载成功则执行，否则结束当前进程。

Join 系统调用：join 的实现个人感觉稍微绕一些，需要理解信号量的作用，下面具体叙述。

父进程调用 join 方法，本质上就是等待相应的子进程结束之后再从 join 中跳出来，也就是把父进程阻塞在 join 中等待相应的子进程结束。这个过程父子进程是通过信号量保持同步的，先判断这个子进程是不是在内核中，若在，则在 join 中新建一个 Semaphore 类，信号量的值初始化为 0，然后把这个信号量插入到父进程的信号量队列中，之后进行父进程进行 p 操作，阻塞，知道子进程结束时在其 Finish 方法中调用相应信号量的 v 操作才让父进程结束 p 操作并跳出，子进程已经结束，父进程删除相应信号量。过程描述的很详细，其中调用的函数或者方法都很简单直观，不再赘述。

三个系统调用的叙述中都未叙述中断处理程序的具体内容，因为这部分内容与实验一相同，都是在寄存器中获取参数，之后调用相关的系统调用或者函数，最后写回结果，故未赘述。

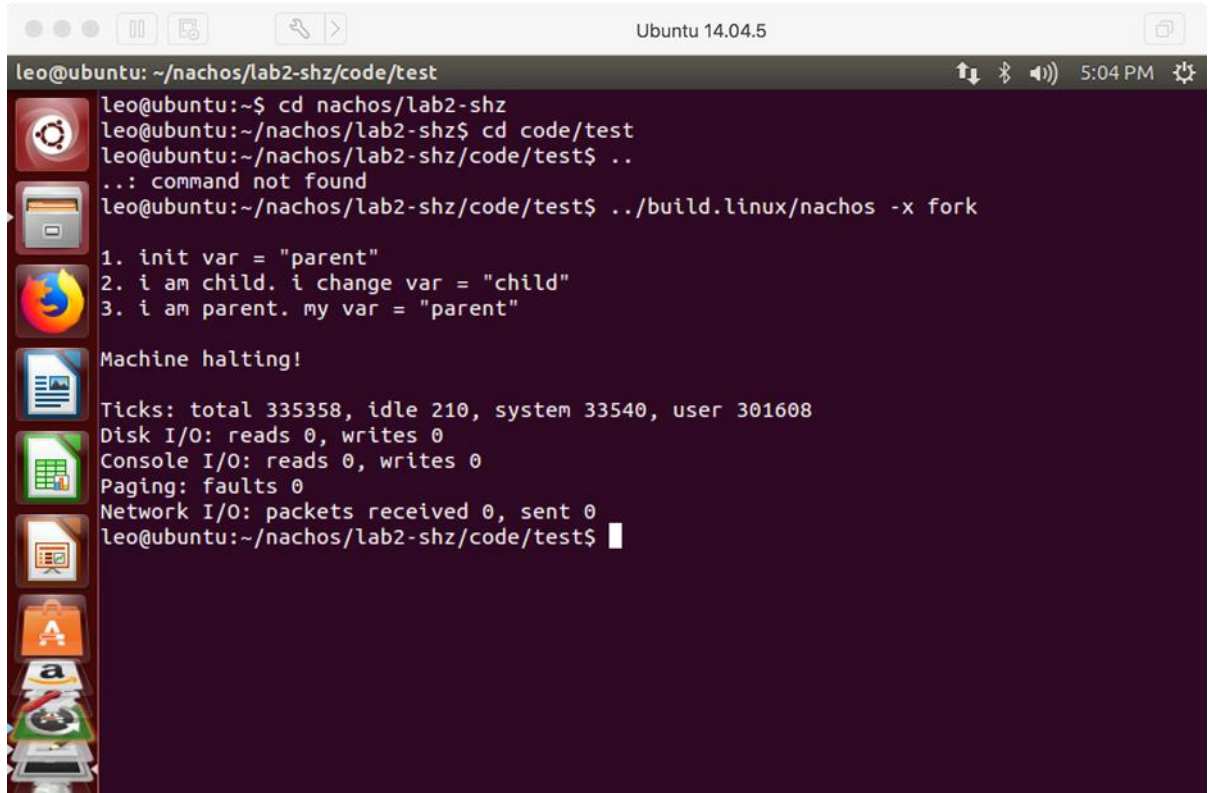
2. 阶段二：编写一个简单的 shell 并实现进程动态优先级调度

shell 的编写：shell 的编写助教已经做了绝大部分工作，如获取指令，分割指令等。

分割后的指令放在 `cmdLine` 数组中，指令数位 `cmdNum`，实现 shell 是调用 `fork` 建立 `cmdNum` 个 shell 的子进程，之后再在 `fork` 返回的子进程（即返回值为 0）中调用 `exec` 执行 `cmdLine`，注意这里都是循环执行，在 `fork` 返回的父进程（即返回值不为 0）中调用 `join`，即等待子进程所以的死进程，亦需要循环调用。

动态优先级调度：我们需要编写的函数是 `Schedule` 类的 `findNextToRun` 方法，方法调用 `flushpriority` 函数动态更新所有就绪队列的优先级以及正在执行的进程的优先级，更新的数值规定助教已经规定，当前进程优先级 `priority=priority -距离上次调度间隔/10`，距离上次调度时间距离是总的时间 `kernel->stats->totalTick`，上次调度时间，`kernel->schedual->lastSwitchTick`，对就绪队列中的进程是每个进程的优先级加上 `AdaptPace` 常数，方法是写了一个接受进程的函数 `flush`，在函数中将优先级加上 `AdaptPace`，之后在 `flushpriority` 中调用 `readyList->Apply(flush)`，对每个进程进程操作。优先级调整后，需要判断是否满足调度条件，即就绪队列不空并且距离上次调度时间不过短，一个简单的判断即可，若不满足调度条件，返回空，即继续执行原来的进程，否则用就绪队列中优先级最大的（`front`）的优先级与当前进程的优先级比较，若就绪队列中大，则进行上下文弹出 `front` 并返回，返回之前要更新调度时间 `lastSwitchTick`，否则返回空。

3. 实验结果：

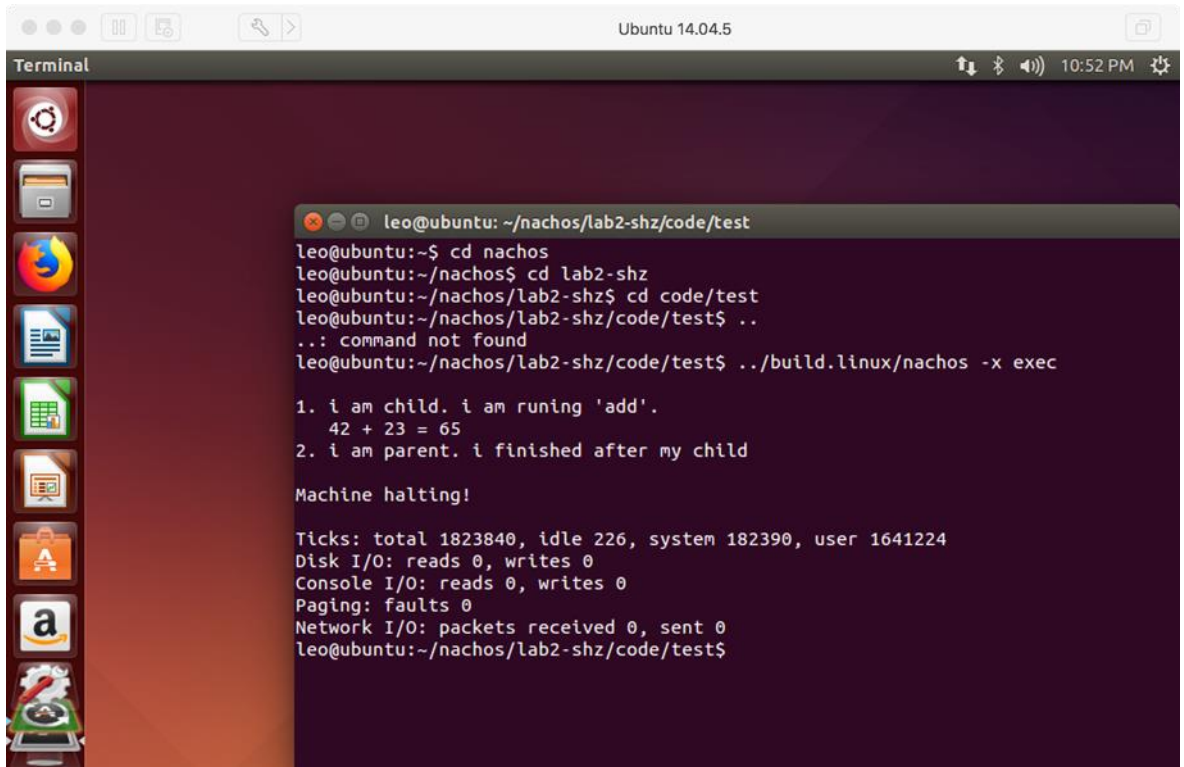


```
leo@ubuntu: ~/nachos/lab2-shz/code/test
leo@ubuntu:~$ cd nachos/lab2-shz
leo@ubuntu:~/nachos/lab2-shz$ cd code/test
leo@ubuntu:~/nachos/lab2-shz/code/test$ ..
...: command not found
leo@ubuntu:~/nachos/lab2-shz/code/test$ ../build.linux/nachos -x fork

1. init var = "parent"
2. i am child. i change var = "child"
3. i am parent. my var = "parent"

Machine halting!

Ticks: total 335358, idle 210, system 33540, user 301608
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
leo@ubuntu:~/nachos/lab2-shz/code/test$
```



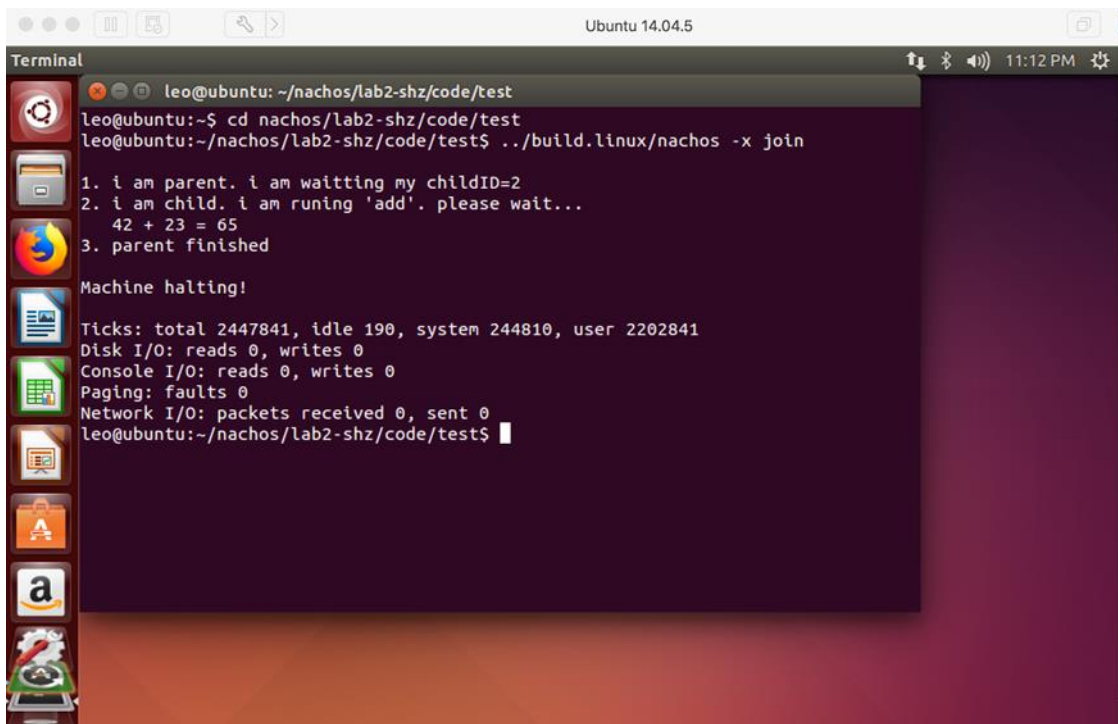
Terminal window showing the execution of a program in the nachos lab2-shz directory. The user leo@ubuntu:~/nachos/lab2-shz/code/test runs the command `../build.linux/nachos -x exec`. The output shows the program's execution, including a child process running 'add' and a parent process finishing after the child. The program halts, and system statistics are displayed.

```
leo@ubuntu: ~/nachos/lab2-shz/code/test
leo@ubuntu:~$ cd nachos
leo@ubuntu:~/nachos$ cd lab2-shz
leo@ubuntu:~/nachos/lab2-shz$ cd code/test
leo@ubuntu:~/nachos/lab2-shz/code/test$ ..
...: command not found
leo@ubuntu:~/nachos/lab2-shz/code/test$ ../build.linux/nachos -x exec

1. i am child. i am runing 'add'.
   42 + 23 = 65
2. i am parent. i finished after my child

Machine halting!

Ticks: total 1823840, idle 226, system 182390, user 1641224
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
leo@ubuntu:~/nachos/lab2-shz/code/test$
```



Terminal window showing the execution of a program in the nachos lab2-shz directory. The user leo@ubuntu:~/nachos/lab2-shz/code/test runs the command `../build.linux/nachos -x join`. The output shows the program's execution, including a parent process waiting for a child process to finish. The program halts, and system statistics are displayed.

```
leo@ubuntu:~/nachos/lab2-shz/code/test
leo@ubuntu:~/nachos/lab2-shz/code/test$ ../build.linux/nachos -x join

1. i am parent. i am waitting my childID=2
2. i am child. i am runing 'add'. please wait...
   42 + 23 = 65
3. parent finished

Machine halting!

Ticks: total 2447841, idle 190, system 244810, user 2202841
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
leo@ubuntu:~/nachos/lab2-shz/code/test$
```

```

running: tid=2    name=./add          status=BLOCKED    pri=223

Ready list contents:
|  |  | tid=0    name=main          status=READY      pri=-419
|  |  |
|  |  |
-----one switch-----
running: tid=2    name=./add          status=BLOCKED    pri=213|
Ready list contents:
|  |  | tid=0    name=main          status=READY      pri=-417
|  |  |
|  |  |
-----one switch-----
running: tid=2    name=./add          status=BLOCKED    pri=193

Ready list contents:
|  |  | tid=0    name=main          status=READY      pri=-415
|  |  |
|  |  |

```

```

leo@ubuntu: ~/nachos/lab2-shz/code/test
leo@ubuntu:~/nachos$ cd lab2-shz
leo@ubuntu:~/nachos/lab2-shz$ cd code
leo@ubuntu:~/nachos/lab2-shz/code$ cd test
leo@ubuntu:~/nachos/lab2-shz/code/test$ ../build.linux/nachos -x shell

nachos >> ./2
i am testshell

nachos >> ./add
42 + 23 = 65

nachos >> ps
Unable to open file ps
exec cmd by Linux
  PID TTY          TIME CMD
  9749 pts/6        00:00:00 bash
  9766 pts/6        00:00:00 nachos
  9768 pts/6        00:00:00 sh
  9769 pts/6        00:00:00 ps

nachos >> ./2; ./add; ./2; ./2
42 + 23 = 65
i am testshell
i am testshell
i am testshell

nachos >>

```

4.实验感想与收获

这次实验由于助教做了大量的工作所以这次实验的代码编写不难，但是要理清整个进程的内容还是有些难度，我的主要时间都花在了阅读 nachos 源码中，对每个类在某个阶段的作用都掌握了（自我感觉），所以虽然代码量不大，但是收获还是很大的。