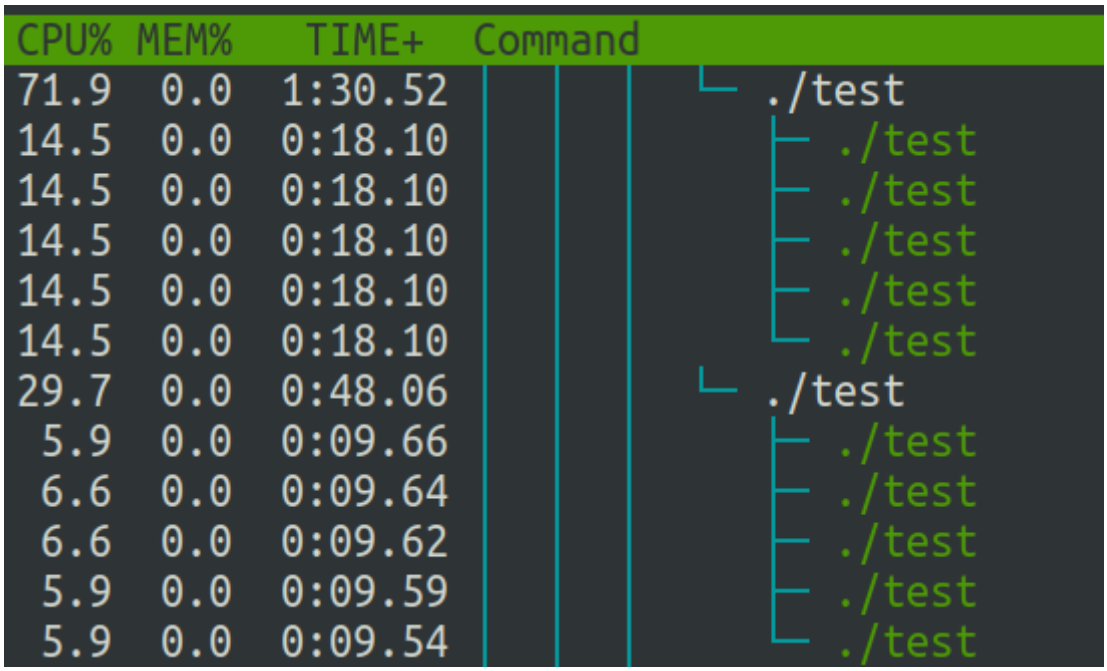
CS353 Project 2

Task 1

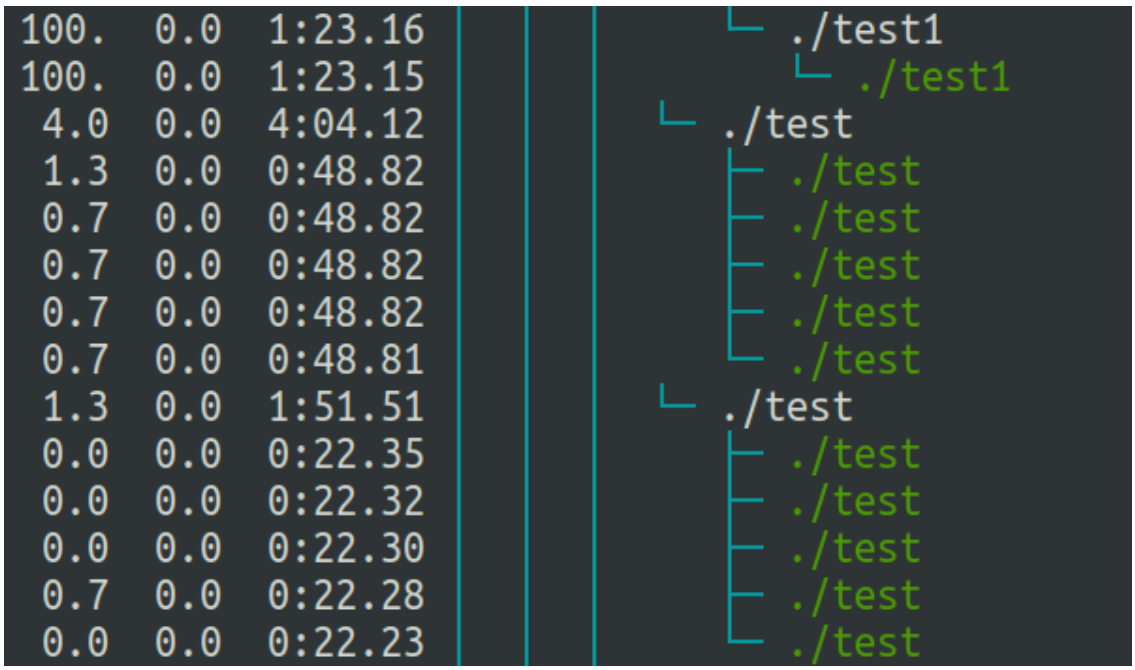
实验内容

1. 创建十个**CPU-bound**程序，并将他们绑定在同一个CPU核心上，修改这些进程(线程)的优先级使得其中5个进程占用大约70%的CPU资源，另外5个进程使用剩下的30%。在同一组中的进程应该具有相同的优先级，使用`top`或`htop`命令验证实验结果

- 

- 上图为结果示例。你可以使用自己编写的程序或任何已有的命令行工具来实现以上效果

2. 在相同的CPU核心上，再创建一个实时进程，验证当这个进程在运行时，会抢占其他十个进程

- 

实验提示

- 使用 `taskset` 将进程绑定到指定CPU核心上
- 创建实时线程需要root权限

提交内容

- 在实验报告描述如何实验过程和思考以及可能遇到的问题
- CPU-bound程序和创建实时进程的源码文件

Task 2

实验内容

修改Linux源代码，为每个进程添加调度次数的记录

具体要求

- 在 `task_struct` 结构体中添加数据成员变量 `int ctx`，用于记录进程的调度次数
- 在进程对应的 `/proc/<PID>` 目录下添加只读文件 `ctx`
- 当读取 `/proc/<PID>/ctx` 时，输出进程当前的调度次数

实验提示

- 进程是如何管理的？阅读Linux源码中 `task_struct` 的定义，在**合适的位置**声明 `ctx` 成员
- 进程是如何被创建的？在进程创建的函数中，在合适的位置初始化 `ctx`，源码文件 `kernel/fork.c`
- 进程是如何被调度的？在调度进程的函数中，找到合适的位置，更新 `ctx`，源码文件 `kernel/sched/core.c`
- 每个进程在 `/proc` 目录下都有自己的目录，那么这些目录是在哪里被创建的？如何自定义文件操作函数？参考 `pid_entry` 和 `tgid_base_stuff`。源码文件 `fs/proc/base.c`
- 这个task需要改动的代码很少，但需要仔细阅读Linux源码
- 测试代码例子

```c
#include <stdio.h>
int main(){
    while(1) getchar();
    return 0;
}
```

提交内容

- 在实验报告中描述修改思路以及具体修改的代码部分（截图或代码）
- 实验过程中遇到的问题以及实验效果

实验提交

提交渠道：Canvas

提交文件：`学号_project2.zip`，源码文件夹 `学号_project2_src`（所有源代码文件以及Makefile），实验报告 `学号_project2_report.pdf`。

实验报告内容包括但不限于实验过程、实验效果截图、实验心得（实验过程中遇到的困难、解决的方法，或者是值得分享的小技巧）。