OS_Lab04

姓名	李懋
学号	2213189
邮箱	2213189@mail.nankai.edu.cn

1 实验题目

- 写一个C/C++程序，列出用户模式下的所有进程

2 实验目的

1. 掌握如何使用C语言编写程序来遍历Linux系统中的 `/proc` 文件系统。
2. 理解Linux系统中进程信息的存储方式。
3. 通过编写程序，模拟 `ps -ef` 命令的功能，列出用户模式下的所有进程。

3 实验原理

1. 在Linux系统中，所有进程的信息都存储在 `/proc` 文件系统中。每个进程都有一个对应的目录，目录名称是进程的PID（进程ID）。在这个目录中，有多个文件存储了进程的各种信息，例如：
    - `status` 文件：包含进程的状态信息，如进程名称、状态、UID等。
    - `cmdline` 文件：包含进程的命令行参数。
2. 通过读取这些文件，可以获取进程的详细信息。
3. 本实验通过遍历 `/proc` 目录，读取每个进程目录中的 `status` 和 `cmdline` 文件，来获取并输出进程的相关信息。
4. C程序具体思路
    1. **遍历目录**：使用 `opendir()`、`readdir()` 和 `closedir()` 函数来遍历 `/proc` 目录。
    2. **读取文件**：使用 `fopen()`、`fgets()` 和 `fclose()` 函数来读取进程目录中的 `status` 和 `cmdline` 文件。
    3. **解析文件内容**：使用 `sscanf()` 函数从文件内容中提取所需的信息，如进程名称、状态和UID。
    4. **输出结果**：将提取的信息格式化输出，类似于 `ps -ef` 命令的结果。
5. 源代码

```
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

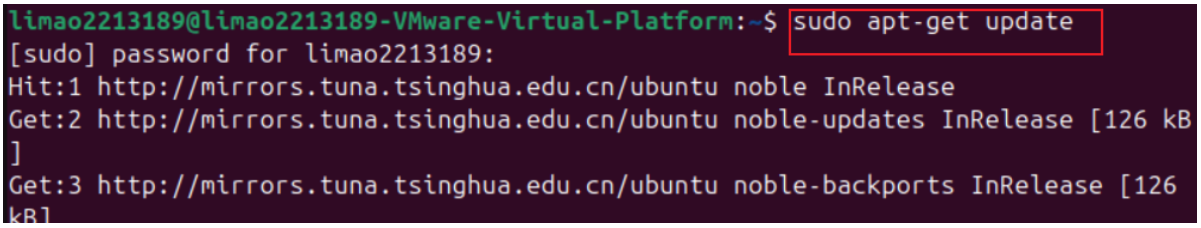
```c
#include <sys/stat.h>
#include <unistd.h>

#define PROC_DIR "/proc"

void print_process_info(const char *pid) {
    char path[256];
    char buffer[1024];
    FILE *fp;

    // Read status file
    snprintf(path, sizeof(path), "%s/%s/status", PROC_DIR, pid);
    fp = fopen(path, "r");
    if (fp == NULL) return;

    char name[256], state[32], uid[32];
    while (fgets(buffer, sizeof(buffer), fp)) {
        if (sscanf(buffer, "Name: %s", name) == 1) continue;
        if (sscanf(buffer, "State: %s", state) == 1) continue;
        if (sscanf(buffer, "Uid: %s", uid) == 1) break;
    }
    fclose(fp);

    // Read cmdline file
    snprintf(path, sizeof(path), "%s/%s/cmdline", PROC_DIR, pid);
    fp = fopen(path, "r");
    if (fp == NULL) return;

    char cmdline[1024] = "";
    fgets(cmdline, sizeof(cmdline), fp);
    fclose(fp);

    // Print process info
    printf("%s\t%s\t%s\t%s\n", uid, pid, state, cmdline);
}

int main() {
    DIR *dir;
    struct dirent *entry;

    dir = opendir(PROC_DIR);
    if (dir == NULL) {
        perror("opendir");
        return 1;
    }

    printf("UID\tPID\tSTATE\tCMD\n");

    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_DIR && atoi(entry->d_name) > 0) {
            print_process_info(entry->d_name);
        }
    }

    closedir(dir);
    return 0;
```

```
}
```

4 实验具体步骤

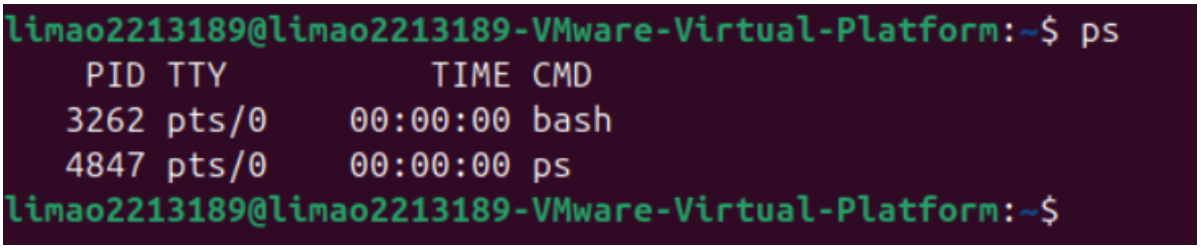
1. 安装相应的工具

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install codeblocks
```



2. IDE这里选择 vscode，之前已经安装好了

3. ps命令



4. ps -ef

```
limao2213189@limao2213189-VMware-Virtual-Platform:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 12:02 ?        00:00:01 /sbin/init splash
root           2       0  0 12:02 ?        00:00:00 [kthreadd]
root           3       2  0 12:02 ?        00:00:00 [pool_workqueue_release]
root           4       2  0 12:02 ?        00:00:00 [kworker/R-rcu_gp]
root           5       2  0 12:02 ?        00:00:00 [kworker/R-sync_wq]
root           6       2  0 12:02 ?        00:00:00 [kworker/R-slub_flushwq]
root           7       2  0 12:02 ?        00:00:00 [kworker/R-netns]
root          10       2  0 12:02 ?        00:00:00 [kworker/0:0H-events_highpri]
root          11       2  0 12:02 ?        00:00:00 [kworker/u512:0-ipv6_addrconf]
root          12       2  0 12:02 ?        00:00:00 [kworker/R-mm_percpu_wq]
root          13       2  0 12:02 ?        00:00:00 [rcu_tasks_kthread]
root          14       2  0 12:02 ?        00:00:00 [rcu_tasks_rude_kthread]
root          15       2  0 12:02 ?        00:00:00 [rcu_tasks_trace_kthread]
root          16       2  0 12:02 ?        00:00:00 [ksoftirqd/0]
root          17       2  0 12:02 ?        00:00:00 [rcu_preempt]
root          18       2  0 12:02 ?        00:00:00 [rcu_exp_par_gp_kthread_worker/1]
root          19       2  0 12:02 ?        00:00:00 [rcu_exp_gp_kthread_worker]
root          20       2  0 12:02 ?        00:00:00 [migration/0]
root          21       2  0 12:02 ?        00:00:00 [idle_inject/0]
root          22       2  0 12:02 ?        00:00:00 [cpuhp/0]
root          23       2  0 12:02 ?        00:00:00 [cpuhp/1]
root          24       2  0 12:02 ?        00:00:00 [idle_inject/1]
root          25       2  0 12:02 ?        00:00:00 [migration/1]
root          26       2  0 12:02 ?        00:00:00 [ksoftirqd/1]
root          27       2  0 12:02 ?        00:00:00 [kworker/1:0-cgroup_destroy]
root          28       2  0 12:02 ?        00:00:00 [kworker/1:0H-events_highpri]
root          29       2  0 12:02 ?        00:00:00 [cpuhp/2]
root          30       2  0 12:02 ?        00:00:00 [idle_inject/2]
root          31       2  0 12:02 ?        00:00:00 [migration/2]
root          32       2  0 12:02 ?        00:00:00 [ksoftirqd/2]
root          34       2  0 12:02 ?        00:00:00 [kworker/2:0H-kblockd]
```

5. ls /proc

```
limao2213189@limao2213189-VMware-Virtual-Platform:/proc$ ls /proc
1     1244  16    2178  230   2448  257   268   2950  3255  461   55   78   98            kallsyms      slabinfo
10    1252  1609  218   2301  245   2571  269   2951  3262  466   56   786  99            kcore         softirqs
100   1254  1613  2185  2306  246   2572  27    296   349   475   57   79   acpi          keys          stat
101   127   1617  219   231   247   2574  270   297   35    476   58   795  asound        key-users     swaps
102   1272  1672  2192  232   2477  2579  271   2977  350   477   59   80   bootconfig    kmsg          sys
103   129   17    2198  233   248   258   2712  298   36    4787  6    804  buddyinfo     kpagecgroup   sysrq-trigger
104   13    170   22    234   249   2581  272   3     3607  4799  606  805  bus           kpagecount    sysvipc
105   130   173   220   235   25    2582  2728  30    37    48    61   806  cgroups       kpageflags    thread-self
106   1305  1751  2209  2359  250   2584  273   301   3775  4839  62   81   cmdline       latency_stats timer_list
107   1309  1769  221   236   2500  2588  2768  3014  3798  4844  63   82   consoles      loadavg       tty
109   131   18    2210  2360  251   259   2779  302   38    4845  64   83   cpuinfo       locks         uptime
11    1323  1823  2214  237   2519  26    28    303   398   4846  65   84   crypto        mdstat        version
110   1335  19    2219  2374  252   260   2811  3036  4     4854  66   85   devices       meminfo       vmallocinfo
111   1350  2     222   238   2528  2606  2826  304   40    49    67   86   diskstats     misc          vmstat
112   1373  20    2222  2389  253   261   2834  3046  4079  5     68   87   dma           modules       zoneinfo
116   14    200   2226  239   2537  2611  2845  3051  41    50    69   88   driver        mounts
117   1419  202   223   2396  2538  2614  2850  3052  418   503   7    89   dynamic_debug mpt
118   146   203   2241  24    254   2616  2858  3071  4325  513   70   90   execdomains   mtrr
1189  1464  207   226   240   2546  262   2864  3085  434   516   71   91   fb            net
119   15    209   2261  241   255   2625  2872  3086  439   517   72   92   filesystems   pagetypeinfo
1193  1572  21    227   242   2550  263   2877  3087  44    519   73   93   fs            partitions
1199  1576  210   2270  243   256   264   29    31    4429  52    74   94   interrupts    pressure
12    1588  211   228   2433  2560  265   2917  3103  45    520   75   95   iomem         schedstat
1223  1589  212   229   244   2564  266   2923  32    4569  53    76   96   ioports       scsi
1226  1590  214   23    2447  2567  267   295   3216  46    54    77   97   irq           self
limao2213189@limao2213189-VMware-Virtual-Platform:/proc$
```

6. 编写一个C语言程序，实现类似的效果
    - 截取了一部分


```
1000    2584    S       /usr/libexec/gsd-sharing
1000    2588    S       /usr/libexec/gsd-smartcard
1000    2606    S       /usr/libexec/gsd-disk-utility-notify
1000    2611    S       /usr/bin/vmtoolsd
1000    2614    S       /usr/libexec/gsd-sound
1000    2616    S       /usr/libexec/gsd-wacom
1000    2625    S       /usr/libexec/evolution-data-server/evolution-alarm-notify
1000    2712    S       /usr/libexec/ibus-memconf
1000    2728    S       /usr/libexec/ibus-extension-gtk3
1000    2768    S       /usr/libexec/ibus-portal
1000    2779    S       /usr/libexec/goa-daemon
1000    2811    S       /usr/libexec/gvfs-udisks2-volume-monitor
1000    2826    S       /usr/libexec/gsd-printer
1000    2834    S       /usr/libexec/goa-identity-service
1000    2845    S       /usr/libexec/evolution-calendar-factory
1000    2850    S       /usr/libexec/gvfs-mtp-volume-monitor
1000    2858    S       /usr/libexec/gvfs-afc-volume-monitor
1000    2864    S       /usr/libexec/gvfs-goa-volume-monitor
1000    2872    S       /usr/libexec/gvfs-gphoto2-volume-monitor
1000    2877    S       /usr/libexec/evolution-addressbook-factory
1000    2917    S       /usr/libexec/ibus-engine-simple
1000    2923    S       /usr/libexec/dconf-service
1000    2950    S       /usr/libexec/gvfsd-metadata
1000    2951    S       /usr/libexec/gvfsd-trash
1000    2977    S       /usr/libexec/gsd-xsettings
1000    3014    S       /usr/bin/gjs
1000    3036    S       /usr/libexec/xdg-desktop-portal
1000    3046    S       /usr/libexec/ibus-x11
1000    3051    S       /usr/libexec/tracker-miner-fs-3
1000    3052    S       /usr/libexec/xdg-desktop-portal-gnome
0       3071    I
0       3085    I
0       3086    I
```

- 列出了和 ps -ef类似的效果，实验成功

5 实验总结

- 通过本次实验，掌握了如何使用C语言编写程序来遍历Linux系统中的 `/proc` 文件系统，并读取其中的进程信息。
    1. **遍历目录**：使用 `opendir()`、`readdir()` 和 `closedir()` 函数来遍历 `/proc` 目录。
    2. **读取文件**：使用 `fopen()`、`fgets()` 和 `fclose()` 函数来读取进程目录中的 `status` 和 `cmdline` 文件。
    3. **解析文件内容**：使用 `sscanf()` 函数从文件内容中提取所需的信息，如进程名称、状态和UID。
    4. **输出结果**：将提取的信息格式化输出，类似于 `ps -ef` 命令的结果。
- 通过这些步骤，不仅理解了Linux系统中进程信息的存储方式，还成功地编写了一个程序来列出用户模式下的所有进程。