OSLab6 Shell

实验报告

吴澄杰　　151220122　　151220122@smail.nju.edu.cn　　2017年6月29日

一、实验进度

此次实验，我完成了 2 个命令，ls 和执行程序。ls 命令可以添加 -a，-l，-h 参数。以 .bin 后缀结尾的文件为可执行文件。例如，仿照Linux，可以通过“./game.bin”命令执行当前目录下的 game 程序。

二、实现细节

1、shell 主体

shell 的主体代码在工程目录下 myshell/myshell.c 中。

```
64 void shell_main()
65 {
66     printf("Welcome to myShell!\n");
67     char cmd[strLength];
68     char arg[argNum][strLength];
69     while (true)
70     {
71         print_prompt();
72         int argc = read_command(cmd, arg);
73         if (!(argc >= 0 && argc <= argNum)) printf("Illegal command!\n");
74         else if (strcmp(cmd, "ls") == 0) cmd_ls(argc, arg);
75         else
76         {
77             char tmp[strLength]; tmp[0] = cmd[0]; tmp[1] = cmd[1]; tmp[2] = '\0';
78             if (strcmp(tmp, "./\0") == 0)
79             {
80                 int i = 2;
81                 while (cmd[i] != '\0') {tmp[i - 2] = cmd[i]; i++;}
82                 tmp[i - 2] = '\0';
83                 if (fork() == 0)
84                 {
85                     if (exec(tmp) == -1)
86                     {
87                         printf("No such file or not executable!\n");
88                         exit();
89                     }
90                 }
91                 else drop_exec();
92             }
93         }
94     }
95     exit();
96 }
```

print_prompt() 为打印提示符。接下来，read_command() 函数会调用 getline() 库函数读出输入的命令（以回车为止），并且切分命令和每一个参数，最后返回参数个数。接下来就根据不同的命令执行。其具体代码在 myshell.c 中可见。

若输入不存在的命令或者参数过多，则会显示“Illegal command!”；若在执行程序时，目标文件不存在或者不是可执行文件，则会显示“No such file or not executable!”。

2、ls 命令

由于实现的文件系统比较简单，文件信息中仅包含文件名、inode号和文件大小，并且尚没有隐藏文件的概念，因此实际上只有 -h，-l 参数会产生效果。-h 参数会输出文件的大小，-l 参数会输出文件的 inode 号。在 myshell.c 中的 cmd_ls 函数是执行 ls 命令的过程。它会准备好参数后进行 ls 系统调用。该系统调用的内核态实现是 kernel/fs/call.c 中的 ls_kernel 函数。

3、执行程序

shell 会先 fork()，随后新产生的 shell 进程会通过 exec() 系统调用执行相应的程序。若该程序不可被执行，那么打印出错误信息后，fork() 出的 shell 会退出。

exec() 的内核态实现是 kernel/process/call.c 中的 exec_kernel() 函数。它以输入的文件名为参数，首先会在文件系统中查找该文件，并检验魔数，验证其为可执行文件。exec 会申请一个新的进程管理块，完成初始化并且继承原进程的必要信息后，释放资源并删除原进程，随后用 load_program 函数将待执行的程序读入内存后，将准备完成的新程序放入就绪队列。

此外，由于我实现了时间片轮转调度，因此，为了防止游戏程序运行时 shell 不停地打扰，我修改了时间片调度的代码，设定为没有其他进程时 shell 才会执行，否则 shell 会一直等待。

当前“磁盘”中提供了 3 个可执行的程序（都在之前的实验中提到过）。其中，game.bin 可以在运行的任意时刻按 q 键退出，sem.bin 可以在运行结束后自动退出。程序退出后，shell 会继续运行。可以继续输入命令。

三、实验心得

我认为 exec() 系统调用是最为关键，也是难度比较大的一环。由于 exec() 的执行过程中涉及到存储管理、进程调度和文件系统的内容，它对之前实现内容的可靠性提出了较高的要求，也考察整个系统的结构是否合理。在实验中，我也经历了较长时间的调试过程。

由于时间比较仓促，我只顾得上实现了程序执行这一比较关键的，其他的命令都没有实现。不过整个实验下来，从运行在硬件上的小游戏开始，逐渐地将操作系统搭成型，并

且添加虚拟存储、文件系统等等内容，现在又可以用 shell 控制程序的执行，还是觉得收获很大。无论是在操作系统的进一步认识上，还是在工程能力的提高上，我认为都受到了之前没有过的训练和提升。