Lab6挑战性任务实验报告

一、实现思路

1. 文件系统功能-create

- user/fsipc.c：`int fsipc_create(const char *path, u_int f_type)`
- include/fs.h：`struct Fsreq_create {};`
- include/fs.h：`#define FSREQ_CREATE 8`
- fs/serv.c：`void serve_create(u_int envid, struct Fsreq_create *rq)`
- fs/serv.c：`serve()`函数中添加相应选项到switch中
- fs/fd.c：`int file_create(char *path, struct File **file)`已提供
- fs/fd.h：`int file_create(...)`函数声明
- user/file.c：`int create(const char *path, u_int f_type)`-用户使用
- 追加诸函数声明到头文件中 user/lib.h

2.添加命令mkdir

```
void umain(int argc, char **argv) {
    int fd = open(argv[1], O_RDONLY);

    if (fd >= 0) {
        fwritef(1, "Directory \"%s\" has already exist!\n", argv[1]);
        return;
    }

    if (create(argv[1], FTYPE_DIR) < 0) {
        fwritef(1, "Failed to create directory!\n");
    }
}
```

touch指令和其类似，即将文件类型改为FTYPE_REG

3.错误命令行不产生panic

将spawn.c中的spawn函数中open文件失败的panic语句删去，调用它时需检验，失败则writef相应提示词

4.添加命令tree

```
# include "lib.h"

# define SPACE_FORMAT "|   "
# define BRANC_FORMAT "|-- "
# define END_FORMAT   "`-- "

char path[MAXPATHLEN];

void traverseTree(char *dir, int layer) {
    struct File file;
```

```
    int fd;
    int i, n;
    char buf[MAXNAMELEN];

    if ((fd = open(dir, O_RDONLY)) < 0) {
        writef("Fail to open directory \"%s\"!", dir);
        exit();
    }

    while ((n = readn(fd, &file, sizeof(struct File)))
                              == sizeof(struct File)) {
        if (file.f_name[0] == 0) continue;

        for (i = 0; i < layer; ++i) fwritef(1, SPACE_FORMAT);
        switch (file.f_type) {
            case FTYPE_DIR:
                fwritef(1, BRANC_FORMAT);
                fwritef(1, "\033[36m%s\033[0m\n", file.f_name);
                strcpy(buf, dir);
                strcat(buf, file.f_name); strcat(buf, "/");
                traverseTree(buf, layer + 1);
                break;

            case FTYPE_REG:
                fwritef(1, BRANC_FORMAT);
                fwritef(1, "%s\n", file.f_name);
                break;

            case FTYPE_BIN:
                fwritef(1, BRANC_FORMAT);
                fwritef(1, "\033[31m%s\033[0m\n", file.f_name);
                break;

            default:
                break;
        }
    }

    if (close(fd) < 0) user_panic("Error: failed to close directory\n");
    if (n) user_panic("Error: unsuccessful read in directory \"%s\"\n", dir);
}

void umain(int argc, char **argv) {
    struct Stat state;

    if (argc == 1) {
        traverseTree("/", 0);
    } else if (argc == 2) {
        strcpy(path, argv[1]);
        if (stat(path, &state) < 0) {
            fwritef(1, "Failed to open \"%s\"!\n", path);
            return;
        }
        if (!state.st_isdir) {
```

```
            fwritef(1, "Command tree can not traverse non-directory file \"%s\"\n",
path);
            return;
        }
        if (path[strlen(path)-1] != '/') strcat(path, "/");
        traverseTree(path, 0);
    } else if (argc == 4) { // with argument
        strcpy(path, argv[1]);
        if (strcmp(argv[2], "-L") == 0) {

        }
    } else {
        fwritef(1, "Wrong format with command \"tree\"!\n");
    }
}
```

5.文件系统功能-增加open的模式-append/create

突然想起来好像是lab5-2的exam内容

- user/lib.h：`#define O_APPEND 0x0004`，`O_CREAT`已存在
- user/file.c：`open()`函数中在得到fd后，将offset置到文件尾即可 => append模式完工
- fs/serv.c：`serve_open()`中添加内容，当fs.c：`file_open()`失败时，判断是否有O_CREAT，若有再调用fs.c：`file_create()`；同时通过这种方式创建出来的文件只能是FTYPE_REG普通型文件 => create模式完工

6.shell功能增添-">"文件重定向时create文件

若>重定向时，右侧文件不存在，则创建该文件。使用上面添加的`O_CREAT`模式即可。

7.完成历史命令功能

首先添加history指令，这个和之前cat类似，只要把.history打印在控制台上即可

```
#include "lib.h"

char buf[8192];

void print_file(int fd) {
    long n;
    int r;

    while((n = read(fd, buf, (long)sizeof buf)) > 0)
        if((r = write(1, buf, n)) != n) {
            fwritef(1, "Error: when writing .history\n");
            exit();
        }
    if(n < 0) user_panic("Error: when reading .history: has read %e", n);
}

void umain(int argc, char **argv) {
    int fd = open(".history", O_RDONLY | O_CREAT);
    if (fd < 0) {
        fwritef(1, "Fail to open file .history\n");
        return;
```

```
    }
    fwritef(1, "==== shell: history ====\n");
    print_file(fd);
    fwritef(1, "\n==== end of history ====\n");
    close(fd);
}
```

然后在sh.c中添加相应机制：

- 在umain中每次读完命令行`readline()`后，将非空命令加入到.history中，集成为函数`save_cmd()`
- 在readline中每次按到特殊键（由于不知道gxemul究竟允许哪些控制字符，反正上下键会直接移动，搞不懂，就设置了Esc和'`'两个分别代表上和下），从.history中读取指定行的命令并放回buf中，控制台上清理一输出字符，集成为函数`get_cmd()`
- 定义`int history_index`，调用sh.c的umain时初始化为0；定义`history_select`，umain中每次循环的开始，将值置为history_index

8.命令行输入支持退格

在user/sh.c的`readline()`函数中，增添`if (buf[i] == 127)`的选项

```
if (buf[i] == 127) { // why not '\b'?
    if (i > 0) {
        fwritef(1, "\033[1D"); // operate cursor
        fwritef(1, "\033[K");
        i -= 2;
    } else i = -1;
}
```

问题就在于，fwritef可能会被时钟中断，导致输出字符被截断，导致达不到预期的删除效果；同时键盘上的退格键竟然对应的是ascii=127，那`\b`是对应什么呢，不是很懂

又增添了一个按键tab，用于当控制台混乱是，查看当前buf中的实际内容

```
if (buf[i] == '\t') {
    buf[i--] = 0;
    writef("[%s]", buf);
}
```

9.实现""功能 ;功能 &功能

修改user/sh.c中的`runcmd()`：

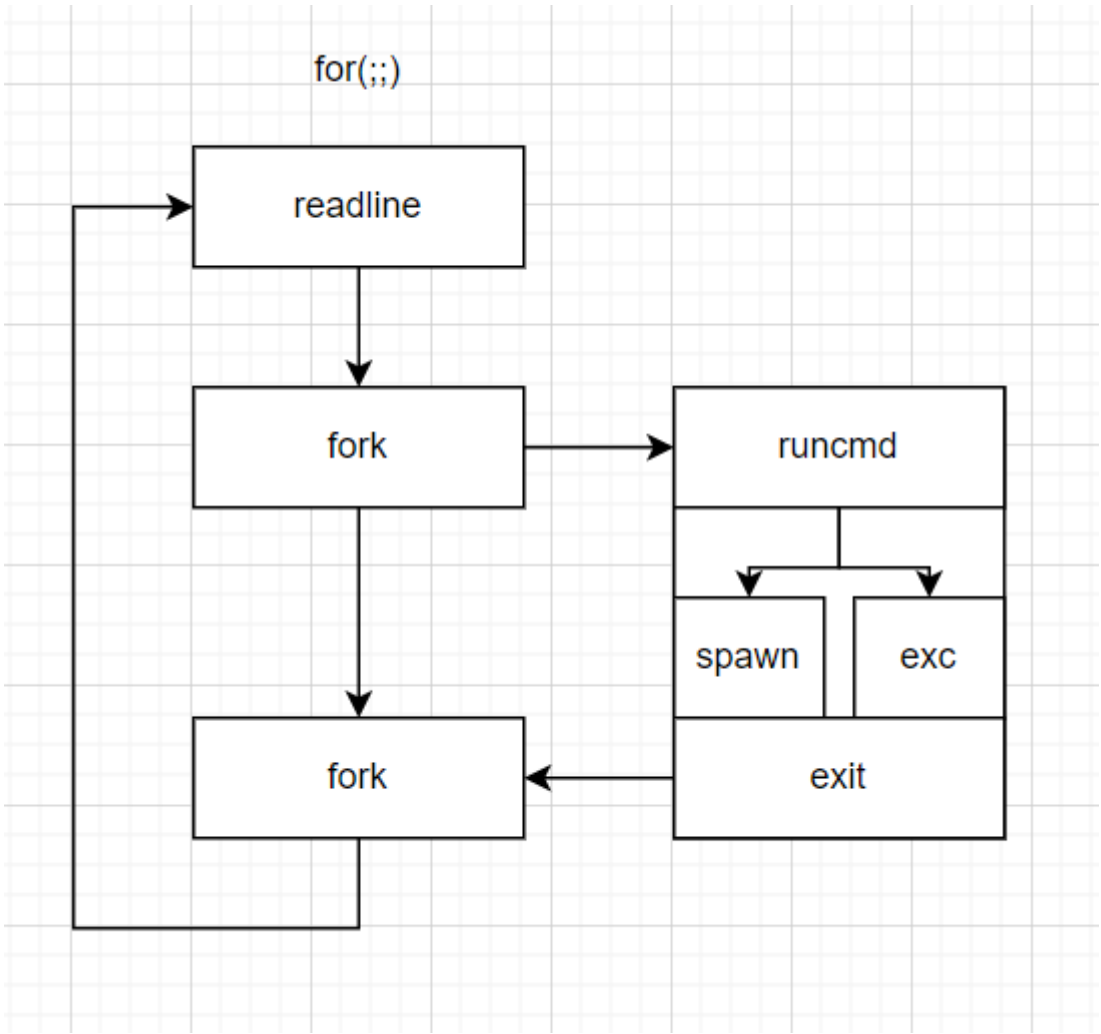
```
case ';':
            envid = fork();
            if (envid == 0) { // child_env
                is_parallel = 0;
                goto runit;
            } else {    // parent_env
                wait(envid);
                argc = 0;
                rightpipe = 0;
                is_parallel = 0;
                do {
```

```
                    close(0);
                    if ((fdnum = opencons()) < 0)
                        user_panic("error in opencons\n");
                } while (fdnum); // continue if fd != 0
                dup(0, 1);
            }
            break;
case '&':
            is_parallel = 1;
            break;
```

修改user/sh.c中的`_gettoken()`:

```
if (*s == '"') {
        s++;
        *p1 = s;
        while (*s && (*s != '"')) s++;
        *s = 0;
        while (*s && !strchr(WHITESPACE SYMBOLS, *s)) s++;
        *p2 = s;
        return 'w';
}
```

10.实现exec函数参与shell进程管理



实现手段，模仿文件系统

添加命令cd 实时修正pwd

艹本以为搞一个全局变量就好，结果发现不同进程之间哪有那么容易搞的

shell总体美观修正

外观相关（用注释或if(debug)的方式减去不必要的输出）：

- fs/ide.c： `ide_write`时输出的diskno
- lib/env.c：`env_free`时的`printf("[%08x] free env %08x\n", curenv ? curenv->env_id : 0, e->env_id)`；env_destroy时的`printf("i am killed ... \n")`
- lib/syscall_all.c：`sys_env_destroy`时的`printf("[%08x] destroying %08x\n", curenv->env_id, e->env_id)`
- mm/pmap.c：`pageout`时新申请完page后的page ins提示信息
- user/spawn.c：产卵时的输出`writef("::::::::::spawn size : %x  sp : %x::::::::\n", size, esp)`
- user/sh.c：第211行，用了debug宏而不是debug_

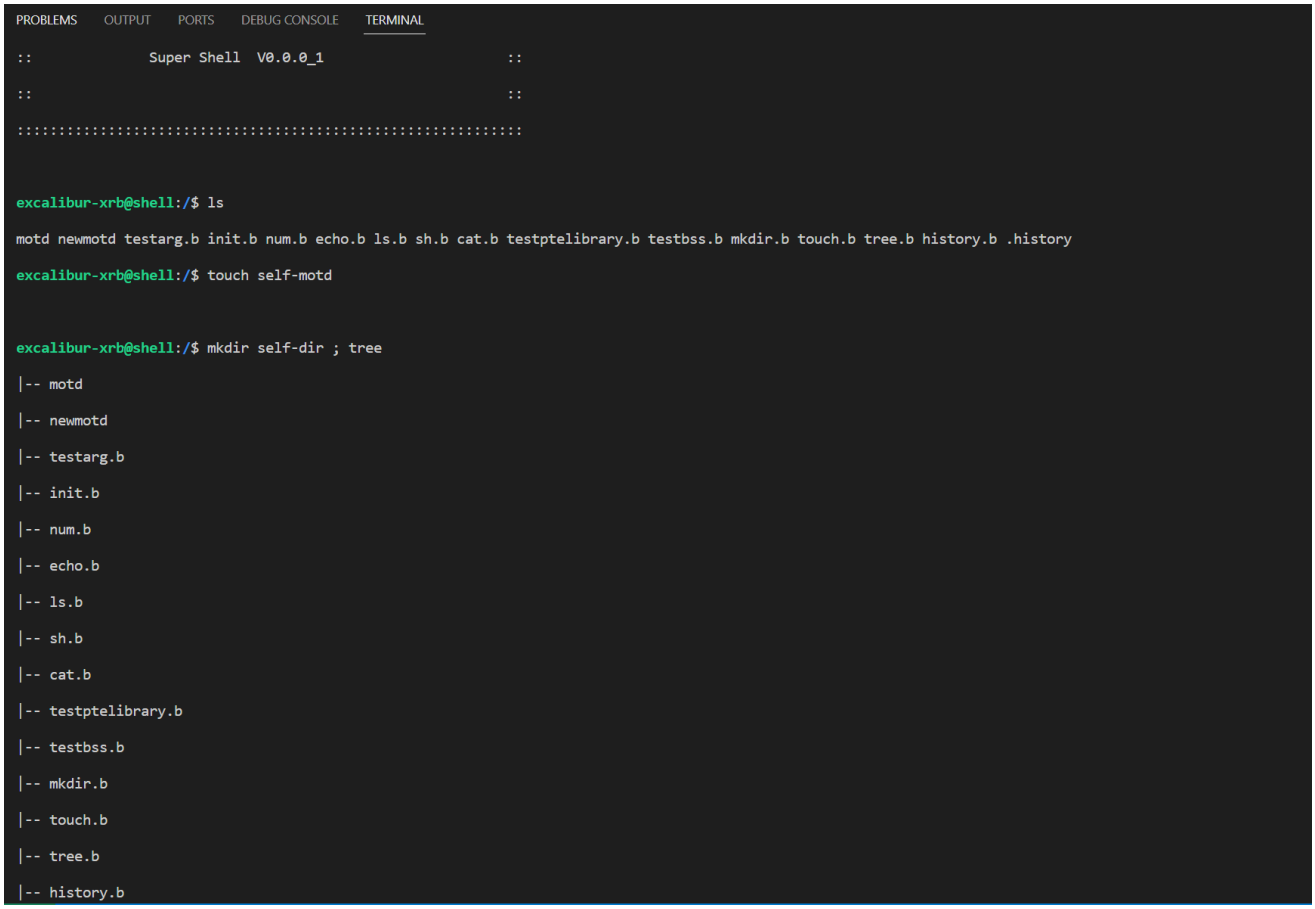
ls命令优化：按文件类型呈现不同颜色

fsformat烧录带有目录结构的fs.img

使用了Linux C库中的dirent.h文件

二、测试样例

基本功能和外观



三、难点记录

难就难在理解OS的一些概念本身：

- shell的运作方式
- 如何构建进程间全局变量（ipc信号传递；共享页面）
- 代码理解和函数使用

我纯纯在照猫画虎。