HW3: OS_6233_HW3_File_System

1. Add functions based on HelloWorld example

2. All the information of processes' id is retrieved from /proc/ directory.

3. The details of one process are retrieved from 'status' in /proc/PROCESS_ID/status file.

Added Functions

1. listdir (#138 ~ #167)

```
// get the content of given direcotry
void listdir(const char *name, int level, void *buf, fuse_fill_dir_t filler) {
    DIR *dir;
    struct dirent *entry;

    if (!(dir = opendir(name)))
        return;
    if (!(entry = readdir(dir)))
        return;

    do {
        if (entry->d_type == DT_DIR) {
            char path[1024];
            int len = snprintf(path, sizeof(path)-1, "%s/%s", name, entry->d_name);
            path[len] = 0;
            if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
                continue;

            // Get the process directory
            if (checkNum(entry->d_name)) {
                filler(buf, entry->d_name, NULL, 0);
                // printf("%*s[%s], %d\n", level*2, "", entry->d_name, checkNum(entry-
>d_name));
            }
            // listdir(path, level + 1);
        }
        else {
            // printf("%*s- %s\n", level*2, "", entry->d_name);
        }
    } while (entry = readdir(dir));
    closedir(dir);
}
```

Get all the content of given directory. I use this function to get all the process folders under /proc/ directory.

2. checkNum

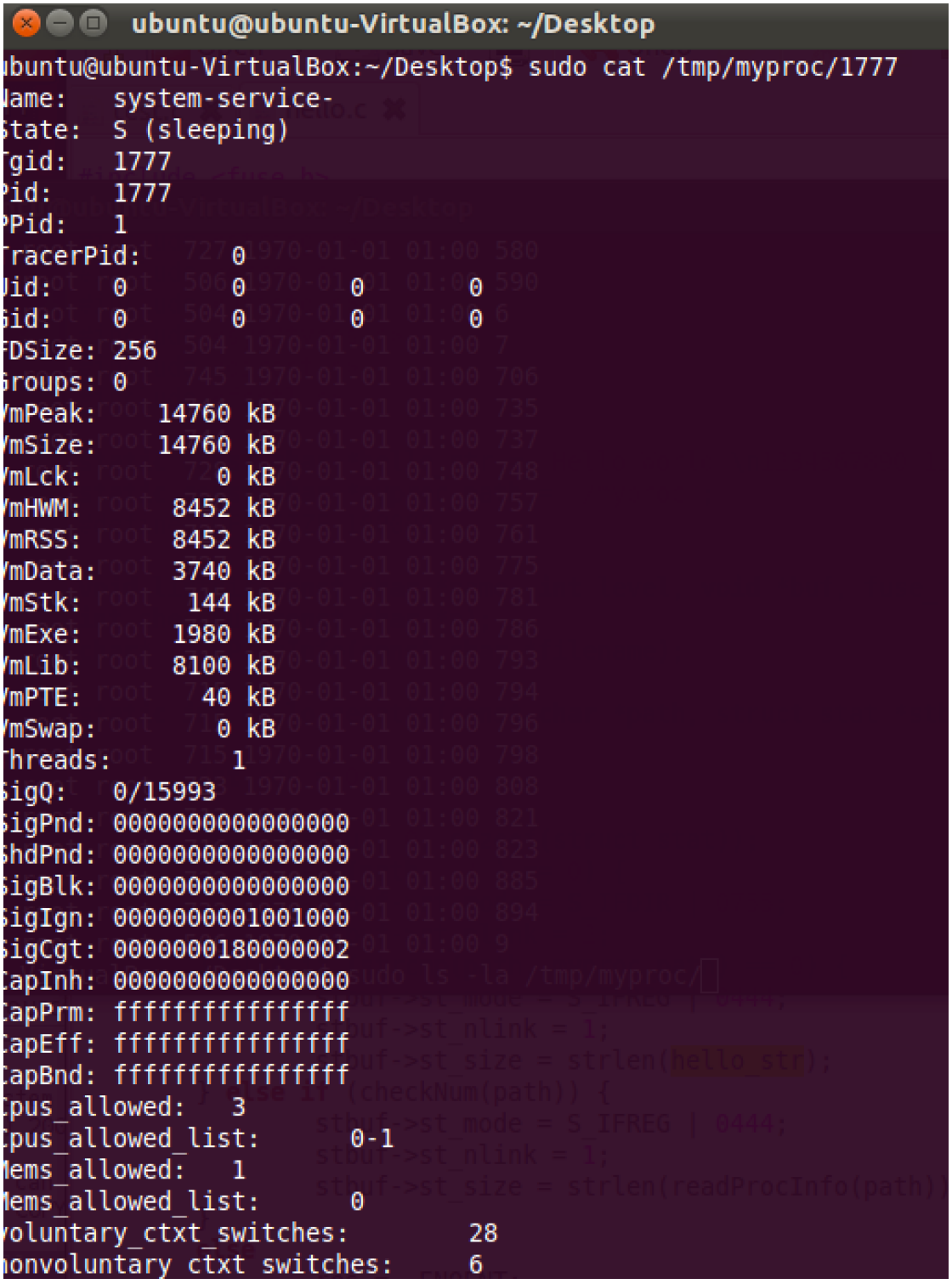
```
// Check if the string is consisted of numbers
int checkNum(const char *p) {
    while(*p) {
        if (*p != '/' && (*p < '0' || *p > '9')) return 0;
        else p++;
    }
    return 1;
}
```

Because the names of processes folder are only consisted of 0~9, by using this function and **listdir**, we can get all the process folders.

3. ReadProcInfo (#179 ~ #207)

```
// read process info
char* readProcInfo(const char *filename) {
    char full_path[100] = "";

    const char *path = "/proc";
    const char *sepreator = "/";
    const char *target = "status";

    strcat(full_path, path);
    strcat(full_path, filename);
    strcat(full_path, sepreator);
    strcat(full_path, target);


    char source[BUFF_SIZE + 1];

    FILE *fp = fopen(full_path, "r");
    if (fp != NULL) {
        size_t newLen = fread(source, sizeof(char), BUFF_SIZE, fp);
        if (newLen == 0) {
            fputs("Error reading file", stderr);
        } else {
            source[++newLen] = '\0'; /* Just to be safe. */
        }

        fclose(fp);
    }

    return source;
}
```

Use readProcInfo function to load the detail info of one process. Here is an example of the output:



Modified Functions

1. hello_getattr

```
static int hello_getattr(const char *path, struct stat *stbuf)
{
        int res = 0;

        memset(stbuf, 0, sizeof(struct stat));
        if (strcmp(path, "/") == 0) {
                stbuf->st_mode = S_IFDIR | 0755;
                stbuf->st_nlink = 2;
        } else if (strcmp(path, myproc_path) == 0) {
                stbuf->st_mode = S_IFREG | 0444;
                stbuf->st_nlink = 1;
                stbuf->st_size = strlen(hello_str);
        } else if (checkNum(path)) {
                stbuf->st_mode = S_IFREG | 0444;
                stbuf->st_nlink = 1;
                stbuf->st_size = strlen(readProcInfo(path));
        }
        else
                res = -ENOENT;

        return res;
}
```

checkNum(path): checking if it is the process folder.
strlen(readProcInfo(path)): getting the size of the file. So we can see the size of each file when we use 'ls -la' to list all the processes file.

2. hello_readdir

```
static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                         off_t offset, struct fuse_file_info *fi)
{

        (void) offset;
        (void) fi;

        if (strcmp(path, "/") != 0)
                return -ENOENT;

        filler(buf, ".", NULL, 0);
        filler(buf, "..", NULL, 0);

        // filler(buf, myproc_path + 1, NULL, 0);

        // char *temp_path = "1143";
        // filler(buf, temp_path + 1, NULL, 0);

        char *proc_path = "/proc/";
        listdir(proc_path, 0, buf, filler);

        return 0;
}
```

listdir(proc_path, 0, buf, filler): after we list all the process, we write the content into buffer, '**buf**'.

3. hello_open

```
static int hello_open(const char *path, struct fuse_file_info *fi)
{
        if (!checkNum(path + 1)) return -ENOENT;
        // if (strcmp(path, myproc_path) != 0) return -ENOENT;

        if ((fi->flags & 3) != O_RDONLY)
                return -EACCES;

        return 0;
}
```

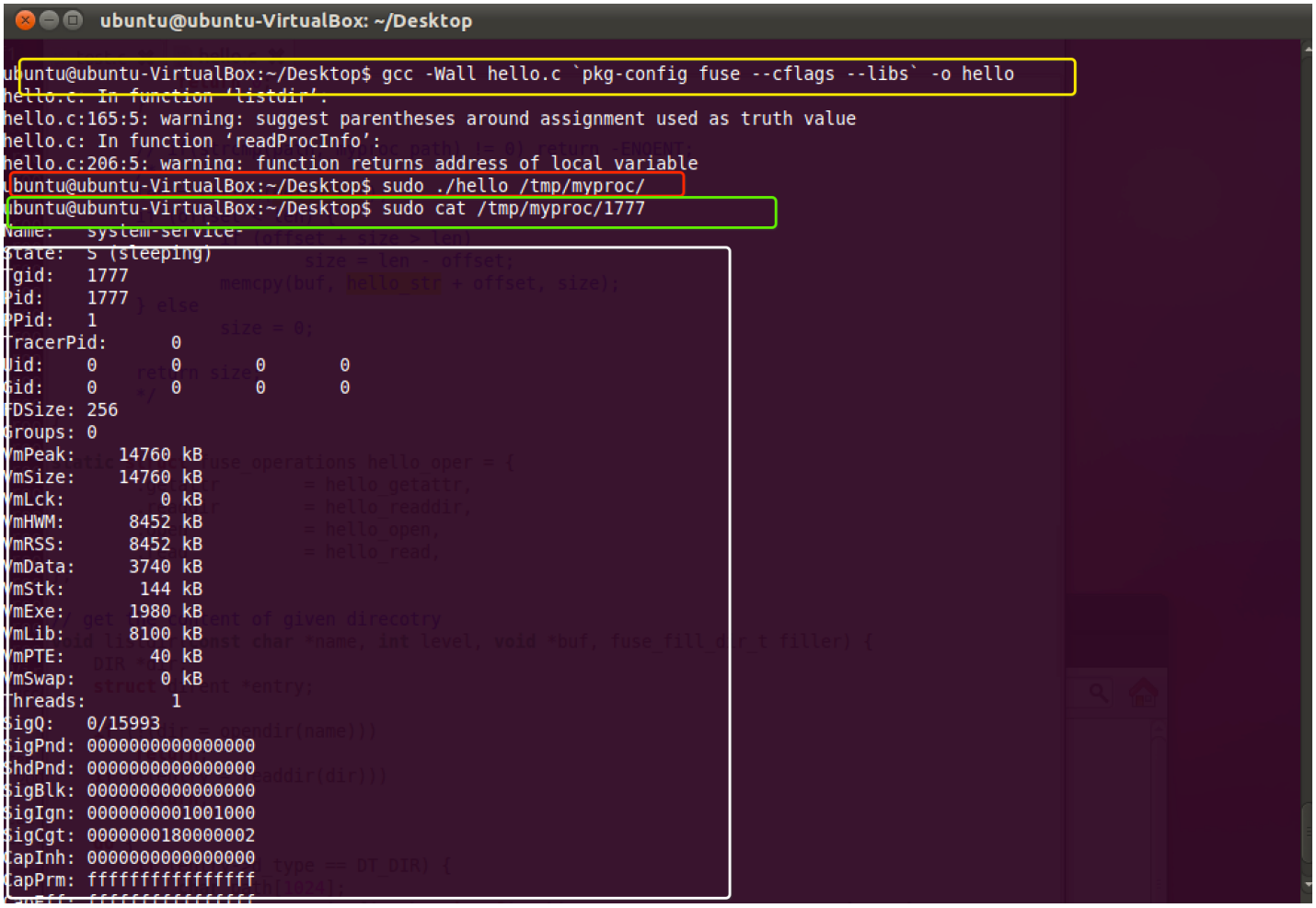
!checkNum(path + 1): if it is not a process folder, we return -ENOENT. By doing this, user can only access the processes folder.

4. hello_read

```
static int hello_read(const char *path, char *buf, size_t size, off_t offset,
                      struct fuse_file_info *fi)
{
        size_t len;
        (void) fi;

        if(checkNum(path + 1)) {
            const char *p = readProcInfo(path);

            len = strlen(p);
            // len = strlen(hello_str);

            if (offset < len) {
                if (offset + size > len)
                        size = len - offset;
                memcpy(buf, p + offset, size);
                // memcpy(buf, hello_str + offset, size);
            }
            else {
                size = 0;
            }

            return size;
        }
        else {
            return 0;
        }
```

We read the process info from ‘proc/PROCESS_ID/status’ file and write the content into buffer.

Compile & Run

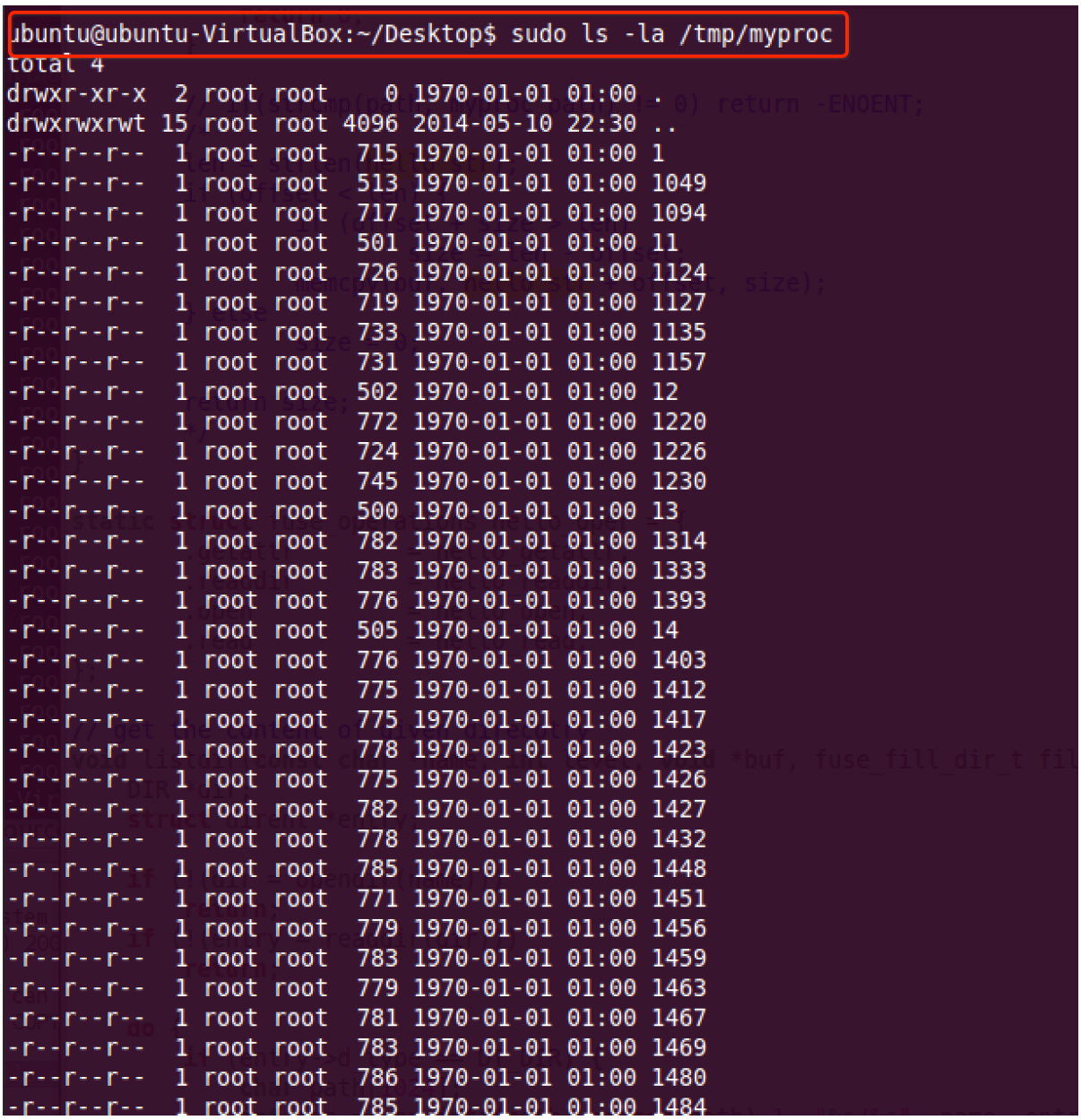


1(yellow): compile source file

2(red): mount the file system to /tmp/myproc/

3(green): read the file(process) 1777

Another screenshot of “ls -la” command



Environment

OS: Ubuntu 11.04 (Linux kernel 2.6.38)
Language: C