BING-HAO LI
bl1861@nyu.edu

OS-HW3-Filesystem

1. Linux version: 11.04
2. Kernel version: 2.6.39
3. Language: C
4. The PIDs come from /proc
5. All the information of the processes comes from /proc/pid/status
6. I create process.c by modifying hello.c
7. Compile: $gcc -Wall process.c `pkg-config fuse --cflags --libs` -o process
8. Run: $./process myproc/

Global variable:

1. The filename is used to store the processes' name in the format "/pid". I assume that there are less than 300 processes in the system and the size of the processes' name will not over 50.
2. The numProc is used to store how many processes in the system.

```
/* the path of process directory */
static const char *proc_path = "/proc";

/* assume total number of file is less than 300 and the path size is less than 50
* filename[n] is /pid;
*/
char filename[300][50];

/* record number of processes in /proc */
int numProc = 0;


```

BING-HAO LI
bl1861@nyu.edu

Functions:

1. int record_pid():

```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
33
34 /* record number of processes in /proc */
35 int numProc = 0;
36
37
38 /* record the PID in the system and return the number of processes in system */
39 int record_pid(){
40         DIR *dir;
41         struct dirent *ent;
42
43         numProc = 0;
44         if((dir = opendir(proc_path)) != NULL){
45                 while((ent = readdir(dir)) != NULL){
46                         int value = atoi(ent->d_name);
47                         if(value == 0){
48                                 continue;
49                         }
50                         if(ent->d_type == DT_DIR){
51                                 strcpy(filename[numProc], ent->d_name);
52                                 numProc++;
53                         }
54                 }
55         }
56         else{
57                 perror("");
58                 return -1;
59         }
60         return numProc;
61 }
62
63 /* to check if *path is /pid */
64 int check_proc(const char *path){
65         int i;
66         for(i = 0; i < numProc; i++){
                                                        37,0-1        17%
```

(1) record_pid() is used to find out all the "/pid" in "/proc" and store them in the filename.

(2) numProc is used to record how many "/pid" are in the system.

(3) I assume that the PIDs start from 1. That is there is no PID = 0 in the system. So, the program can use atoi[line 46] to tell whether the filename is number or not.

2. int check_proc(const char *path):

BING-HAO LI
bl1861@nyu.edu

```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
60          return numProc;
61 }
62
63 /* to check if *path is /pid */
64 int check_proc(const char *path){
65          int i;
66          for(i = 0; i < numProc; i++){
67                  if(strcmp(path, filename[i]) == 0){
68                          return 0;
69                  }
70          }
71          return 1;
72 }
73
74 /* If the *path is the path of a process, return the size of the process status*/
75 size_t fsize(const char *path){
76          FILE *fp;
77          char str[50];
78          char content[100000];
79          sprintf(str, "%s%s%cstatus", proc_path, path, 47);
80          size_t len = 0;
81          fp = fopen(str, "r");
82          if(fp != NULL){
83                  len = fread(content, sizeof(char), 100000, fp);
84                  fclose(fp);
85          }
86          else{
87                  printf("Can't open %s\n", str);
88          }
89          return len;
90 }
91
92
93 static int hello_getattr(const char *path, struct stat *stbuf)
                                                                    93,1          31%
```

(1) It is used to check whether the *path is one of the "/pid" in the system or not. If the path is one of the "/pid" in the system, return 0. Otherwise, return 1.

3. size_t fsize(const char *path):

```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
65          int i;
66          for(i = 0; i < numProc; i++){
67                  if(strcmp(path, filename[i]) == 0){
68                          return 0;
69                  }
70          }
71          return 1;
72 }
73
74 /* If the *path is the path of a process, return the size of the process status*/
75 size_t fsize(const char *path){
76          FILE *fp;
77          char str[50];
78          char content[100000];
79          sprintf(str, "%s%s%cstatus", proc_path, path, 47);
80          size_t len = 0;
81          fp = fopen(str, "r");
82          if(fp != NULL){
83                  len = fread(content, sizeof(char), 100000, fp);
84                  fclose(fp);
85          }
86          else{
87                  printf("Can't open %s\n", str);
88          }
89          return len;
90 }
91
92
```

(1) It read the size of the file "/proc/pid/status" and return the size. If the program can't read the file, it return the size 0 (I assign it as 0 at [line 80].)

BING-HAO LI
bl1861@nyu.edu

Fuse_operations:

1. hello_getattr

```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
 93 static int hello_getattr(const char *path, struct stat *stbuf)
 94 {
 95         int res = 0;
 96
 97         memset(stbuf, 0, sizeof(struct stat));
 98
 99         if (strcmp(path, "/") == 0) {
100                 stbuf->st_mode = S_IFDIR | 0755;
101                 stbuf->st_nlink = 2;
102         }
103         else if (check_proc(path + 1) == 0){
104                 size_t len = fsize(path);
105
106                 /* set process read_only */
107                 stbuf->st_mode = S_IFREG | 0444;
108                 stbuf->st_nlink = 1;
109                 stbuf->st_size = len;
110         }
111         else
112                 res = -ENOENT;
113
114         /* set the total blocks of process*/
115         stbuf->st_blocks = stbuf->st_size % 1024 == 0? stbuf->st_size / 1024: stbuf->st_size / 1024 + 1;
116
117         return res;
118 }
119
120
121
122 static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
123                          off_t offset, struct fuse_file_info *fi)
124 {
125         (void) offset;
                                                                    119,0-1       48%
```

(1) [line 103] the program uses function check_proc() to tell the path is the path of a process file.

(2) [line 104] the program gets the size of status file by fsize().

(3) [line 107] the program sets all process files in /my_proc read_only.

(4) [line 115] the program sets the how many blocks in file.

2. hello_readdir

```
122 static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
123                          off_t offset, struct fuse_file_info *fi)
124 {
125         (void) offset;
126         (void) fi;
127
128         if (strcmp(path, "/") != 0)
129                 return -ENOENT;
130
131         filler(buf, ".", NULL, 0);
132         filler(buf, "..", NULL, 0);
133
134         /* put PIDs in the filler */
135         int num = record_pid();
136         int i;
137         for(i = 0; i < num; i++){
138                 filler(buf, filename[i], NULL, 0);
139         }
140
141         return 0;
142 }
143
```

(1) When system read myproc/ directory, the program checks the processes in "/proc" and store the PIDs in the filename by function record_pid() [line 135] and put all the PIDs in the directory by filler[line 137-139].

BING-HAO LI
bl1861@nyu.edu

3. hello_open

```
145 static int hello_open(const char *path, struct fuse_file_info *fi)
146 {
147         /* user can only open /pid */
148         if (check_proc(path + 1) != 0)
149                 return -ENOENT;
150
151         if ((fi->flags & 3) != O_RDONLY)
152                 return -EACCES;
153
154         return 0;
155 }
156
```

(1) [line 148-149] the program checks the path whether it is process's path or not. If not, it returns –ENOENT.

4. hello_read()

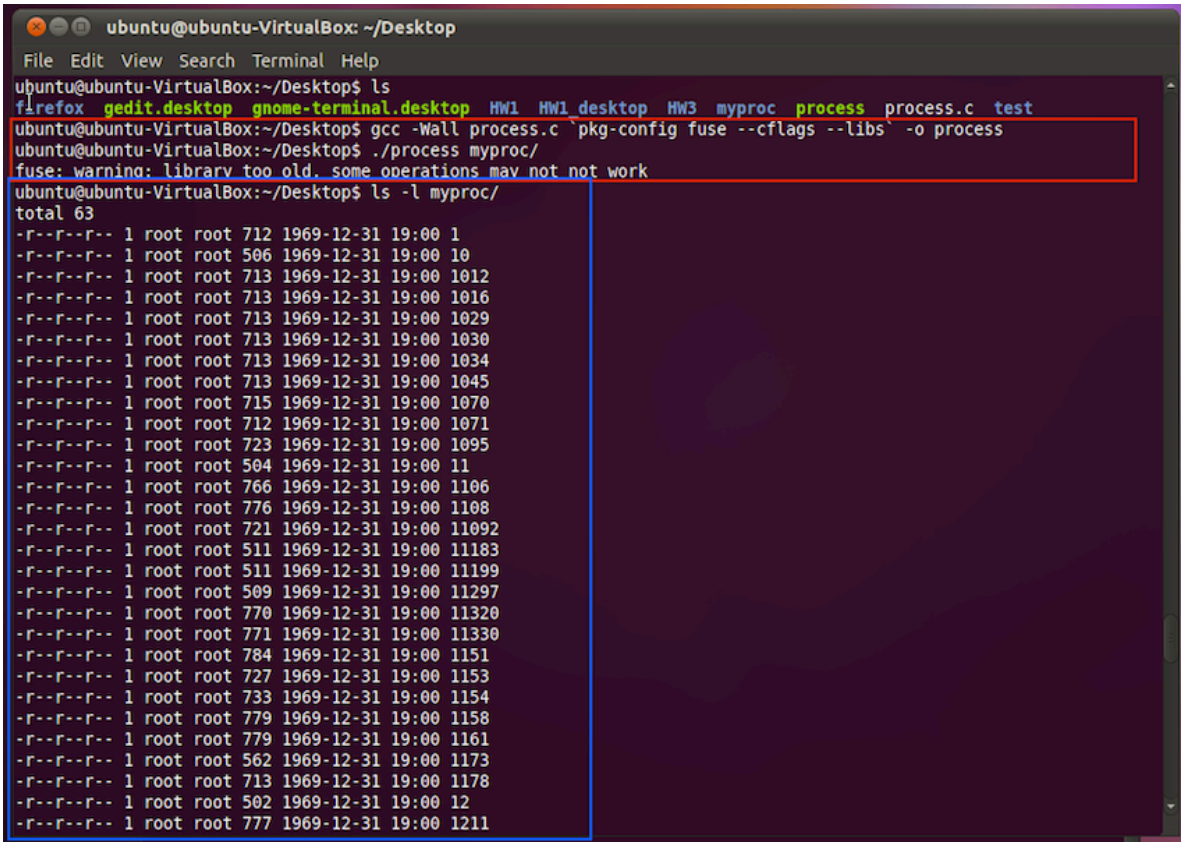
```
157 static int hello_read(const char *path, char *buf, size_t size, off_t offset,
158                       struct fuse_file_info *fi)
159 {
160         size_t len;
161
162         (void) fi;
163
164
165         if(check_proc(path + 1) == 0){
166
167                 /* assume the length of path is less than 50 */
168                 char path_buffer[50] = "";
169                 sprintf(path_buffer, "%s%s%cstatus", proc_path, path, 47);
170
171                 FILE *fp;
172                 fp = fopen(path_buffer,"r");
173                 if(fp != NULL){
174                         size_t length = 1000;
175                         char *line = NULL;
176                         int read;
177
178                         /* assume size of line in file is less than 10000 */
179                         char content[10000] = "";
180                         while((read = getline(&line, &length, fp)) != -1){
181                                 printf("%s\n", line);
182                                 strcat(content, line);
183                         }
184
185                         len = strlen(content);
186                         if (offset < len) {
187                                 if (offset + size > len)
188                                         size = len - offset;
189                                 memcpy(buf, content + offset, size);
190                         }
191                         else
192                                 size = 0;
                                                                    192,1-8       83%
```

(1) the program reads /proc/pid/status [line 180-183] and copy the content into buffer [line 189]

BING-HAO LI
bl1861@nyu.edu

Result:

Figure.1



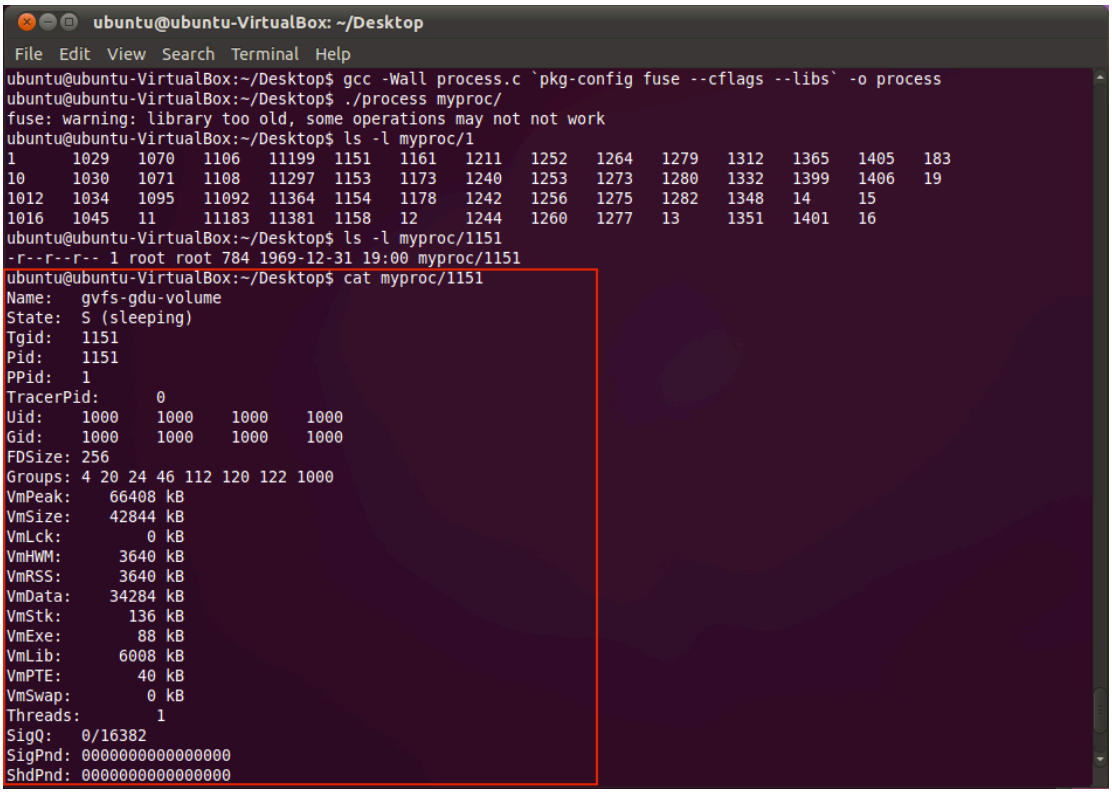
```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
ubuntu@ubuntu-VirtualBox:~/Desktop$ ls
firefox  gedit.desktop  gnome-terminal.desktop  HW1  HW1_desktop  HW3  myproc  process  process.c  test
ubuntu@ubuntu-VirtualBox:~/Desktop$ gcc -Wall process.c `pkg-config fuse --cflags --libs` -o process
ubuntu@ubuntu-VirtualBox:~/Desktop$ ./process myproc/
fuse: warning: library too old, some operations may not not work
ubuntu@ubuntu-VirtualBox:~/Desktop$ ls -l myproc/
total 63
-r--r--r-- 1 root root 712 1969-12-31 19:00 1
-r--r--r-- 1 root root 506 1969-12-31 19:00 10
-r--r--r-- 1 root root 713 1969-12-31 19:00 1012
-r--r--r-- 1 root root 713 1969-12-31 19:00 1016
-r--r--r-- 1 root root 713 1969-12-31 19:00 1029
-r--r--r-- 1 root root 713 1969-12-31 19:00 1030
-r--r--r-- 1 root root 713 1969-12-31 19:00 1034
-r--r--r-- 1 root root 713 1969-12-31 19:00 1045
-r--r--r-- 1 root root 715 1969-12-31 19:00 1070
-r--r--r-- 1 root root 712 1969-12-31 19:00 1071
-r--r--r-- 1 root root 723 1969-12-31 19:00 1095
-r--r--r-- 1 root root 504 1969-12-31 19:00 11
-r--r--r-- 1 root root 766 1969-12-31 19:00 1106
-r--r--r-- 1 root root 776 1969-12-31 19:00 1108
-r--r--r-- 1 root root 721 1969-12-31 19:00 11092
-r--r--r-- 1 root root 511 1969-12-31 19:00 11183
-r--r--r-- 1 root root 511 1969-12-31 19:00 11199
-r--r--r-- 1 root root 509 1969-12-31 19:00 11297
-r--r--r-- 1 root root 770 1969-12-31 19:00 11320
-r--r--r-- 1 root root 771 1969-12-31 19:00 11330
-r--r--r-- 1 root root 784 1969-12-31 19:00 1151
-r--r--r-- 1 root root 727 1969-12-31 19:00 1153
-r--r--r-- 1 root root 733 1969-12-31 19:00 1154
-r--r--r-- 1 root root 779 1969-12-31 19:00 1158
-r--r--r-- 1 root root 779 1969-12-31 19:00 1161
-r--r--r-- 1 root root 562 1969-12-31 19:00 1173
-r--r--r-- 1 root root 713 1969-12-31 19:00 1178
-r--r--r-- 1 root root 502 1969-12-31 19:00 12
-r--r--r-- 1 root root 777 1969-12-31 19:00 1211
```

(1)The read square in figure.1 show how to compile and run the program

(2)The blue square in figure.1 display the output of $ls –l myproc/. As we can see, it shows all processes' name and their attributions.

BING-HAO LI
bl1861@nyu.edu

Figure.2



```
ubuntu@ubuntu-VirtualBox: ~/Desktop
File  Edit  View  Search  Terminal  Help
ubuntu@ubuntu-VirtualBox:~/Desktop$ gcc -Wall process.c `pkg-config fuse --cflags --libs` -o process
ubuntu@ubuntu-VirtualBox:~/Desktop$ ./process myproc/
fuse: warning: library too old, some operations may not not work
ubuntu@ubuntu-VirtualBox:~/Desktop$ ls -l myproc/1
1      1029   1070   1106   11199  1151   1161   1211   1252   1264   1279   1312   1365   1405   183
10     1030   1071   1108   11297  1153   1173   1240   1253   1273   1280   1332   1399   1406   19
1012   1034   1095   11092  11364  1154   1178   1242   1256   1275   1282   1348   14     15
1016   1045   11     11183  11381  1158   12     1244   1260   1277   13     1351   1401   16
ubuntu@ubuntu-VirtualBox:~/Desktop$ ls -l myproc/1151
-r--r--r-- 1 root root 784 1969-12-31 19:00 myproc/1151
ubuntu@ubuntu-VirtualBox:~/Desktop$ cat myproc/1151
Name:   gvfs-gdu-volume
State:  S (sleeping)
Tgid:   1151
Pid:    1151
PPid:   1
TracerPid:      0
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
FDSize: 256
Groups: 4 20 24 46 112 120 122 1000
VmPeak:    66408 kB
VmSize:    42844 kB
VmLck:         0 kB
VmHWM:      3640 kB
VmRSS:      3640 kB
VmData:    34284 kB
VmStk:       136 kB
VmExe:        88 kB
VmLib:      6008 kB
VmPTE:        40 kB
VmSwap:        0 kB
Threads:        1
SigQ:   0/16382
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
```

(1) The red square in figure.2 shows the information of 1151 process when the system read it.

Reference:

FUSE

http://sourceforge.net/p/fuse/wiki/Hello_World/

http://www.cs.nmsu.edu/~pfeiffer/fuse-tutorial/html/

Stat

http://man7.org/linux/man-pages/man2/stat.2.html

O_RDONLY

http://lxr.free-electrons.com/source/include/uapi/asm-generic/fcntl.h#L19