گزارش آزمایش دوم درس سیستم عامل

سیدعلی سادات اخوانی ۸۱۰۱۹۳۴۲۵

محمدحسین نوروزی ۸۱۰۱۹۳۴۹۹

محمد رضایی ۸۱۰۱۹۱۵۱۳

کارهایی که برای انجام این پروژه انجام دادیم:

- اضافه کردن ۳ سیستم‌کال گفته شده به هسته‌ی لینوکس
- پیاده‌سازی نحوه‌ی `iterate` بر روی تمامی `process` های در حال اجرای سیستم
- پیاده‌سازی پیداکردن آدرس `path` فایل مربوط به `process`
- پیاده‌سازی نحوه‌ی کار با `linked list` هسته‌ی لینوکس و استفاده ازماکروها و توابع آن
- پیاده‌سازی سیستم `free` کردن حافظه
- نوشتن کد تست برای سیستم‌کال‌ها و کدهای زده شده
- پاسخ پرسش‌ها

مراحل اضافه‌کردن `System Call` به هسته‌ی لینوکس:

```
1. Modify the architecture specific sources to point to your syscall code.
2. Modify the generic source files to point to your syscall code.
3. Create the code for our system call.
4. Make sure the kernel will compile the code for our syscall.
5. Creating a custom syscall.

appending the line: .long sys_hello_world at the end of
$linux/arch/x86/kernel/syscall_table_32.S

appending the line #define __NR_hello_world 333 after the line #define
__NR_inotify_init1 332 in the file $linux/arch/x86/include/asm/unistd_32.h.
```

iterate بر روی تمامی process های در حال اجرای سیستم

```
for_each_process(task)
```

پیاده‌سازی پیداکردن آدرس path فایل مربوط به process

```
struct files_struct *current_files;
struct fdtable *files_table;
unsigned int *fds;
 int i=0;
 struct path files_path;
 char *cwd;
 char *buf = (char *)kmalloc(GFP_KERNEL,100*sizeof(char));
current_files = current->files;
    files_table = files_fdtable(current_files);
while(files_table->fd[i] != NULL)
 files_path = files_table->fd[i]->f_path;
 cwd = d_path(&files_path,buf,100*sizeof(char));
printk(KERN_ALERT "Open file with fd %d  %s", i,cwd);
```

پیاده‌سازی نحوه‌ی کار با linked list هسته‌ی لینوکس و استفاده ازماکروها و توابع آن

```
list_for_each(p, &(gb_file_array[hash_key]->list))

INIT_LIST_HEAD(&(gb_file_array[j]->list));

file_entity_init(task->pid, i, cwd, new_node);

list_add(&new_node->list, cur_head);
```

نوشتن کد تست برای سیستم‌کال‌ها و کدهای زده شده

```
#define sys_init_hash_table      339
#define sys_show_pid_fd          337
#define sys_free_hash_table      338
syscall(sys_init_hash_table, size);
syscall(sys_show_pid_fd, sag);
syscall(sys_free_hash_table);
```

پاسخ پرسش‌ها

سوال ۱:

0	stdin	Standard Input
1	stdout	Standard Output
2	stderr	Standard Error

سوال ۲:

۳ مورد مختلف برای تمام عملیات‌های IO وجود دارد. File, pipe, socket

سوال ۳:

مسیر /dev/null تمامی دیتاهایی که در آن می‌نویسیم را دور می‌ریزد اما در انتهای کار به ما می‌گوید که عملیات درست انجام شده یا نه. معمولاً برای دور ریختن output stream های یک process به کار می‌آید یا ایجاد شرایط مناسب برای input stream

مسیر /dev/ttyX یک فایل است که نمایانگر terminal پردازش فعلی است. به عنوان مثال اگر در لینوکس چندین ترمینال باز کنیم و دستور tty را در آنها بزنیم، عد خروجی که نمایش داده می‌شود متفاوت است. نکته‌ی دیگر این‌که اگر دستور

echo 1 ->/dev/tty001 را بزنیم، در ترمینال مربوط به آن پراسس ۱ را نشان می‌دهد