

CSIS0230A Principle of Operating Systems (Class A)

Tutorial 6

Page Table

In this tutorial, you are required to add a system call by installing a module. The system call, of number 250, is as follows:

```
asm linkage int sys_getpt(pid_t pid, unsigned long addr, char *buf);
```

Given a specific process *pid* and its linear address *addr*, fill *buf* with the page table that contains the address *addr*.

Input:

pid—the specified process ID

addr—the specified linear address

buf—the buffer provided by caller to store the page table. Its size should be 4K.

Return:

–*ESRCH*—process can not be found

–*EFAULT*—memory specified by *buf* is not writable

1—the page table containing the linear address *addr* is not in memory

0—success

After you finish writing the system call. Compile it and install the module.

In the next page, there is a program that tests your system call. Understand it, run it, and compare the result against the content of `/proc/pid/maps`, where *pid* is the process id of any running process (you can get a list by the command `ps ax`). Then remove the module, rerun the program and see the result.

Hints:

1. To **compile** a module called `modname.c`, use:

```
gcc -Wall -c -O2 modname.c
```

To **install** the module, use:

```
insmod modname.o
```

To **uninstall** the module, use:

```
rmmmod modname
```

2. Remember to perform security check, by using `copy_to_user(void *to, void *from, int size)` to access the memory buffer provided by the user.
3. Use `printk()` for debugging.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <cstring>
#include <cerrno>
#include <cstdlib>
#include <syscall.h>
#include <sys/types.h>
using namespace std;

#define __NR_getpt 250
__syscall3(int, getpt, pid_t, p, unsigned long, laddr, int *, buf);

void dump(int buf[1024]) {
    string line;
    cout << hex;
    getline(cin, line);
    for (int i = 0; i < 1024; i+=8) {
        if (i > 0 && i % (16*8) == 0) {
            cout << "Press enter";
            getline(cin, line);
        }
        cout << setw(4) << i << ':';
        for (int j = 0; j < 8; ++j)
            cout << ' ' << setw(8) << buf[i+j];
        cout << endl;
    }
}

int main(int argc, char * argv[])
{
    if (argc != 2) {
        cout << "Usage: " << argv[0] << " pid" << endl;
        return 1;
    }
    char *ret;
    pid_t pid = strtol(argv[1], &ret, 10); // base 10, first failed char in ret
    if (*argv[1] == 0 || *ret) {
        cout << "Argument must be a number." << endl;
        return 1;
    }
    for (;;) {
        unsigned long addr;
        cout << "Address in hex? (-1 ends) ";
        cin >> hex >> addr;
        if (!cin || addr == (unsigned long)-1)
            break;
        int buf[1024];
        switch (getpt(pid, addr, buf)) {
            case -1:
                cerr << "Error: " << strerror(errno) << endl;
                return 1;
            case 1:
                cout << "Page not present." << endl;
                break;
            case 0:
                dump(buf);
        }
    }
}
```