

# CSIS0230A Principle of Operating Systems (Class A)

## Tutorial 5

### Process Structure

In this tutorial, you have to add two system calls—*getchild* and *getname*. The specifications of the system calls are as follows:

*asmlinkage* **int** *sys\_getchild*(*pid\_t* *pid*, *pid\_t\** *child*, **int** *size*)

Given a specific process *pid*, return the process IDs of all its children in the *child* array of size *size*. At most that many entries in *child* will be used by the system call. It returns the number of children of *pid*. Thus a caller can check whether the buffer is large enough by inspecting the return value.

Input:

*pid*—the specified process ID  
*child*—for storing the children process IDs  
*size*—the size of *child* array

Return:

non-negative—the number of children of *pid*  
-ESRCH—process cannot be found  
-EFAULT—memory specified in *child* is not writable

*asmlinkage* **int** *sys\_getname*(*pid\_t* *pid*, **char\*** *name*)

Given a process *pid*, return the name of the program executed by the process.

Input:

*pid*—the specified process ID  
*name*—a buffer for storing the *name* of the process

Return:

0—the request is served successfully  
-ESRCH—process cannot be found  
-EFAULT—memory specified in *name* is not writable

After you finish writing these system calls, write a user program to test whether the system calls work. A *pid* should be read from the command line, and the program should print all the *pids* and names of all its children.

For example, one can execute the program as

```
./a.out 541
```

Output:

```
The children of pid 541 is the following
9305: in.telnetd
9146: in.telnetd
```

One possible structure of the program:

```
#include <syscall.h>
#include <errno.h>
#include <sys/types.h>
#include <stdio.h>
#define __NR_getchild 191
#define __NR_getname 192
_syscall3(int, getchild, pid_t, pid, pid_t*, child, int, size);
_syscall2(int, getname, pid_t, pid, char*, name);
int main(int argc, char** argv) {
    pid_t pid;
    pid = atol(argv[1]);
    /* find all pid's children process IDs */

    /* for each child process ID
       find the name of process
       print the process ID and name */
}
```

### Hints:

1. There are some data structures or functions that you may need to use:

<b>function or struct</b>	<b>include</b>
<i>task_struct</i>	<i>&lt;linux/sched.h&gt;</i>
<i>pid_t</i>	<i>&lt;linux/types.h&gt;</i>
<i>copy_from_user(void* to, void* from, int size)</i>	<i>&lt;asm/uaccess.h&gt;</i>
<i>copy_to_user(void* to, void* from, int size)</i>	<i>&lt;asm/uaccess.h&gt;</i>
<i>for_each_task(task_struct)</i>	<i>&lt;linux/sched.h&gt;</i>

2. Using the user address space without checking will leave a really big security hole in your kernel.
3. Don't forget to include *<linux/kernel.h>*.
4. Use *printk* for debugging. To see the messages from *printk*, run your program from the console rather than from an X terminal.
5. Don't forget to modify *entry.S* and *Makefile*.