# COMP 3500 Introduction to Operating Systems

# Project 5 – Processes and System Calls

# Adding System Calls to OS/161

**Objectives:**

* To add a system call into OS/161
* To write a user program to test the new system call
* To run and test the user program on OS/161

**1. Configuration**

You run the following commands to configure the source code tree for the Linux machine on which you are working.

%cd ~/cs161/src

%./configure

You can configure a kernel named ASST2 using the following command:

%cd ~/cs161/src/kern/conf

%./config ASST2

**2. System Call Implementation**

The following steps demonstrate how to implement a sample system call getpid.

**2.1 Create a System Call Implementation File**

Your system call implementation files (e.g., file\_syscalls.c and proc\_syscalls.c) should be residing in the OS/161 kernel. We place all the system call implementation files in the following directory:

~/cs161/src/kern/userprog

You need to create a system call implementation file named getpid\_syscall.c in the above directory. The sample implementation file is given below:

#include <types.h>

#include <syscall.h>

#include <thread.h>

#include <curthread.h>

/\* Sample implementation of sys\_getpid() \*/

int

sys\_getpid(pid\_t \*retval)

{

\*retval = curthread->t\_pid;

return 0;

}

You also need to update struct thread in kern/include/thread.h by adding the following data item:

pid\_t t\_pid;

Note that this is only a sample source code file. In this project, you should place file related system calls in file\_syscalls.c and process related system calls in proc\_syscalls.c

**2.2 Update Configuration File and Reconfigure the Project**

Now you can update the configuration file (i.e., conf.kern) located in src/kern/conf

The following line should be added into src/kern/conf/conf.kern

file userprog/getpid\_syscall.c

Now you reconfigure the project (see Section 1 for details), which will be rebuilt in the next step (see Section 2.5).

**2.3 Declare Prototype of sys\_getpid**

The prototype of sys\_getpid may be included in the following file:

~/cs161/src/kern/include/syscall.h

Add the following function prototype in the above file:

int sys\_getpid(pid\_t \*retval);

**2.4 Update the system call handler syscall.c**

The system call handler syscall.c is located in the following directory:

~/cs161/src/kern/arch/mips/mips

You must modify syscall.c in such a way the system call request of getpid issued by user programs can be handled by the sys\_getpid() function, which we implemented in Step 2.1.

**Important!** The following code segment should be added in the switch-case statement of the mips\_syscall() function in syscall.c

### case SYS\_getpid:

### err = sys\_getpid(&retval);

### break;

**2.5 Rebuild the OS/161 Kernel**

Follow the commands below to rebuild the kernel.

%cd ~/cs161/src/kern/compile/ASST2

%make depend

%make

%make install

### **3. Test System Calls**

**3.1 Create a User Program for the New System Call**

We place all the test programs in the following directory:

~/cs161/src/testbin

Each test program and its associated files (e.g., Makefile) are organized in a dedicated directory. For example, test program forktest.c and its Makefile can be found in:

~/cs161/src/testbin/forktest

In what follows, let us use forktest as a template to create a test driver for the getpid system call.

**Step 1:** Create a new directory using forktest as a template:

%cd ~/cs161/src/testbin

%cp –r forktest getpidtest

**Step 2:** Change source code name:

%cd getpidtest

%mv forktest.c getpidtest.c

**Step 3:** **Important!** Modify getpidtest.c as follows. This program is quite simple; it calls the getpid system call and then shuts down OS/161.

#include <unistd.h>

#include <stdio.h>

int main() {

int mypid;

mypid = getpid();

reboot(RB\_REBOOT);

return 0;

}

**Step 4:** Modify Makefile and depend.mk by replacing forktest with getpidtest

**Step 5:** Compile getpidtest.c using cs161-gcc. This can be down through running Makefile as below.

%make

The make utility program compile getpidtest.c and generate an execute file called getpidtest

**Step 6:** Copy the executable file getpidtest into ~/cs161/root/testbin

%cp getpidtest ~/cs161/root/testbin/getpidtest

The above executable file will be loaded by OS/161 through the p command in the main menu.

**3.2 Run the User Program in OS/161**

You can follow the instructions below to run the testing program created in Step 3.1:

%cd ~/cs161/root

%./sys161 kernel

**Important!** In the menu prompt type:

p /testbin/getpidtest