CS356 **Operating System Projects** **Spring 2018**

Project 1: Android Process Tree

Objectives:

- Install and use Android Virtual Devices.
- Install NDK, cross compile the program and run it on AVD.
- Effectively use Linux system calls for process control and management.
- Familiarize task_struct
- Concurrent execution of processes.

Make sure your system is 64-bits Linux system.

Problem Statement:

1. **Install Android Virtual Device (AVD), and create a new AVD.**

   The links of necessary files are given in [1].

   The location of JDK and SDK is up to you. Finally, create the AVD named as "OsPrj-StudentID", make target as "Android 6.0-API Level 23". If your system is 64-bit Linux, [2] should to be considered.

2. **Install Android NDK, run HelloWorld in your AVD.**

   The link of NDK is given in [1]. You should download it, and extract it to a proper location.

   Add the location of NDK to Environment Variables [4] so that we can use "ndk-build" in

other directory.

Make a directory for HelloWorld project and write a “HelloWorld!” program. The files structure should be:

-Helloworld

-JNI

-HelloWorld.c

-HelloWorld.h

-Android.mk

The content of Android.mk is like this:

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_SRC_FILES := hello.c
LOCAL_MODULE := helloARM
LOCAL_CFLAGS += -pie -fPIE
LOCAL_LDFLAGS += -pie -fPIE
LOCAL_FORCE_STATIC_EXECUTABLE := true
include $(BUILD_EXECUTABLE)
```

More information about Android.mk is in [3].

If you do not familiar with the Android.mk, please do not change everything but project name in the Android.mk

Finally, use adb to debug the HelloWorld on AVD.

3. **Write a new system call in Linux**

In android system, we can use *ps* to see the information of all process, but we cannot use *pstree* to see the relationship of those process intuitively like what we can do in Linux. So we need a new system call. The system call you write should take two arguments and return the process tree information in a depth-first-search (DFS) order. The prototype for your system call will be:

```
int ptree(struct prinfo *buf, int *nr);
```

You should define struct prinfo as:

```
struct prinfo {
        pid_t parent_pid;       /* process id of parent, set 0 if it has no parent*/
        pid_t pid;              /* process id */
        pid_t first_child_pid;  /* pid of youngest child, set 0 if it has no child */
        pid_t next_sibling_pid;  /* pid of older sibling, set 0 if it has no sibling*/
        long state;             /* current state of process */
```

```
        long uid;           /* user id of process owner */
        char comm[64];      /* name of program executed */
};
```

The argument *buf* points to a buffer for the process data, and *nr* points to the size of this buffer (number of entries). The system call copies as many entries of the process tree data to the buffer as possible, and stores the number of entries actually copied in *nr*.

The original Android kernel does not support module. Therefore, you should use the kernel we supported online [6]. You can learn how to start AVD with a new kernel in [7].

Your system call should return the total number of entries on success (this may be bigger than the actual number of entries copied).

4. Test your new system call

Write a simple C program which calls *ptree*. Your program should print the entire process tree (in DFS order) using tabs to indent children with respect to their parents. The output format should be:

```
printf(/* correct number of \t */);
```

printf("%s,%d,%ld,%d,%d,%d,%d\n", p.comm, p.pid, p.state, p.parent_pid,

p.first_child_pid, p.next_sibling_pid, p.uid);

5. Test *ptree*

Generate a new process, output ("*StudentID*Parent is %d", pid).

Then generates its child process, output ("*StudentID*Child is %d", pid).

Use *execl* to execute *ptree* in the child process，show the relationship between above two process.

6. Caesar Encryption Sever

Caesar cipher is one of the simplest and most widely known encryption techniques. During encryption, each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. In this problem, we set the number=3.

For example,

Plain: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Cipher: `DEFGHIJKLMNOPQRSTUVWXYZABC`

Plain: `abcdefghijklmnopqrstuvwxyz`

Cipher: `defghijklmnopqrstuvwxyzabc`

Please develop a Caesar Encryption Server, which receives plaintext from clients and sends the corresponding ciphertext to clients.

Implementation Details:

1. **Only the letters** need to be encrypted, e.g. **How are you? → Krz duh brx?**
2. The Server can serve at most **2 clients** concurrently, more clients coming have to wait.
3. The server-side program must be **concurrent multi-threaded**. Pay attention, not multi-process!
4. Client inputs ***:q*** to end the service.
5. For simplicity, you can execute one server and multiple clients in **one host**.
6. Before you start this problem, I **highly recommend** you to read these two pages:
   - http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
   - http://www.linuxhowtos.org/C_C++/socket.htm

   All you need about multi-thread and network programming in Linux can be found in the pages above.
7. We provide the code framework of Server-side and Client-side for you (See PPT).
8. Only **2** source files are needed: *client.c* and *server.c*.
9. For your test, please start server-side program first, then start client-side program!

Material to be submitted:

1. Compress the source code of the programs into **Prj1+StudentID.tar** file. It contains all *.c, *.h and Android.mk files. Use meaningful names for the file so that the contents of the file are obvious. Enclose a README file that lists the files you have submitted along with a one sentence explanation. Call it **Prj1README**.

2. Only internal documentation is needed. Please state clearly the purpose of each program at the start of the program. Add comments to explain your program. (-5 points, if insufficient.)
3. Test runs: It is very important that you show that your program works for all possible inputs. Submit online a single typescript file clearly showing the working of all the programs for correct input as well as graceful exit on error input.
4. Send your **Prj1+StudentID.tar** file to cs356.sjtu@gmail.com.
5. Due date: **Apr. 19, 2018**, submit on-line **before midnight**.

Appendix

[1] Official Download

JDK: www.oracle.com/technetwork/java/javase/downloads/index.html

SDK: http://developer.android.com/sdk/index.html#Other

NDK: http://developer.android.com/ndk/downloads/index.html#download

The complete version (no need to download anything from Google) we supply:

SDK:http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-sdk-linux.tar.gz

NDK:http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-ndk-r11-linux-x86_64.zip

[2] If your Linux is 64-bits, execute the following command before setting up your AVD

sudo apt-get install libc6:i386 libgcc1:i386 gcc-4.6-base:i386 libstdc++5:i386 libstdc++6:i386

[3] http://developer.android.com/ndk/guides/android_mk.html

[4] How to add location to Environment Variables.

Add following sentence in ~/.bashrc(for common user) or /etc/profile(for root user):

export PATH=#I absolute path you want#:$PATH

Then type source ~/.bashrc or source /etc/profile in terminal.

DO NOT change any other value in Environment Variables

[5] Some useful adb command:

To check the AVD status:

adb devices

To move a file to the emulator:

adb push #source path ~/hello/hello.o# #target path on device /data/misc#

To use shell on Android:

adb shell

Then you can use shell command like linux.

To pull a file out of the emulator:

adb pull #source path in device# #target path#

More commands about adb:

adb help

[6] kernel: http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-kernel.tar.gz

[7] Start AVD with customized kernel:

emulator –avd YourAvdName –kernel KernelLocation –show-kernel

YourAvdName could be OsPrj

KernelLocation could be ~/kernel/goldfish/arch/arm/boot/zImage

-show-kernel makes kernel information shown in your shell.

[8] A fast pass to download necessary resources for this project:

URL: http://pan.baidu.com/s/1nuSvYh7

Password：ned2