Project 1: Android Process Tree

Fan Wu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Spring 2021

Objectives

- Install and use Android Virtual Devices (AVD)
- Install Native Development Kit (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

Enviroment

- Implementation
  - AVD(Android Virtual Devices)
    - SDK version r24.4.1
- Development
  - Linux（64-bits）
    - Ubuntu (recommended)
    - Debian
    - Fedora
  - VMware

Resources

- Programming in C/UNIX System Calls and Subroutines using C
  - http://www.cs.cf.ac.uk/Dave/C/CE.html
- Posix Thread Programming
  - https://computing.llnl.gov/tutorials/pthreads/
- Android SDK Install
  - http://developer.android.com/sdk/installing/index.html?pkg=tools
- Android.mk
  - http://developer.android.com/ndk/guides/android_mk.html

Install JDK

- JDK is Java SE Development Kit which is necessary for android project.
- You can download it for your system at:
  - http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Don't forget to modify your Environment Variables.
  - For Windows, just run the .exe file. Every thing will be done automatically.
  - For Linux, add these to ~/.bashrc

```
export JAVA_HOME=/usr/lib/jdk1.8.0_73
export JRE_HOME=/usr/lib/jdk1.8.0_73/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

Install SDK

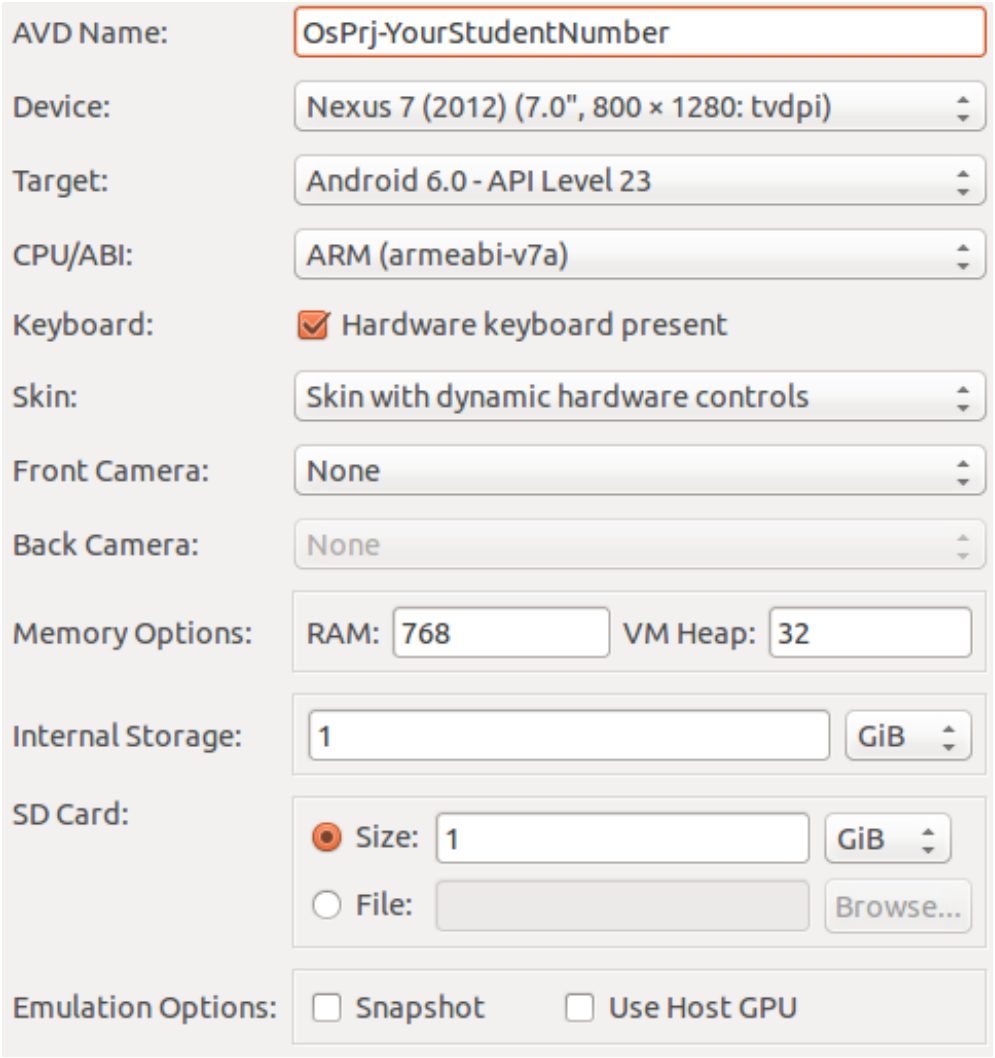
- For Windows
  - http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-sdk-windows.7z
- For Linux
  - http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-sdk-linux.tar.gz
- Extract them into a proper location.

Set Up AVD

- For Windows
  - Double click “AVD Manager.exe”
- For Linux
  - Execute ./tools/android avd in SDK folder.

- The recommended configuration of AVD is on next page

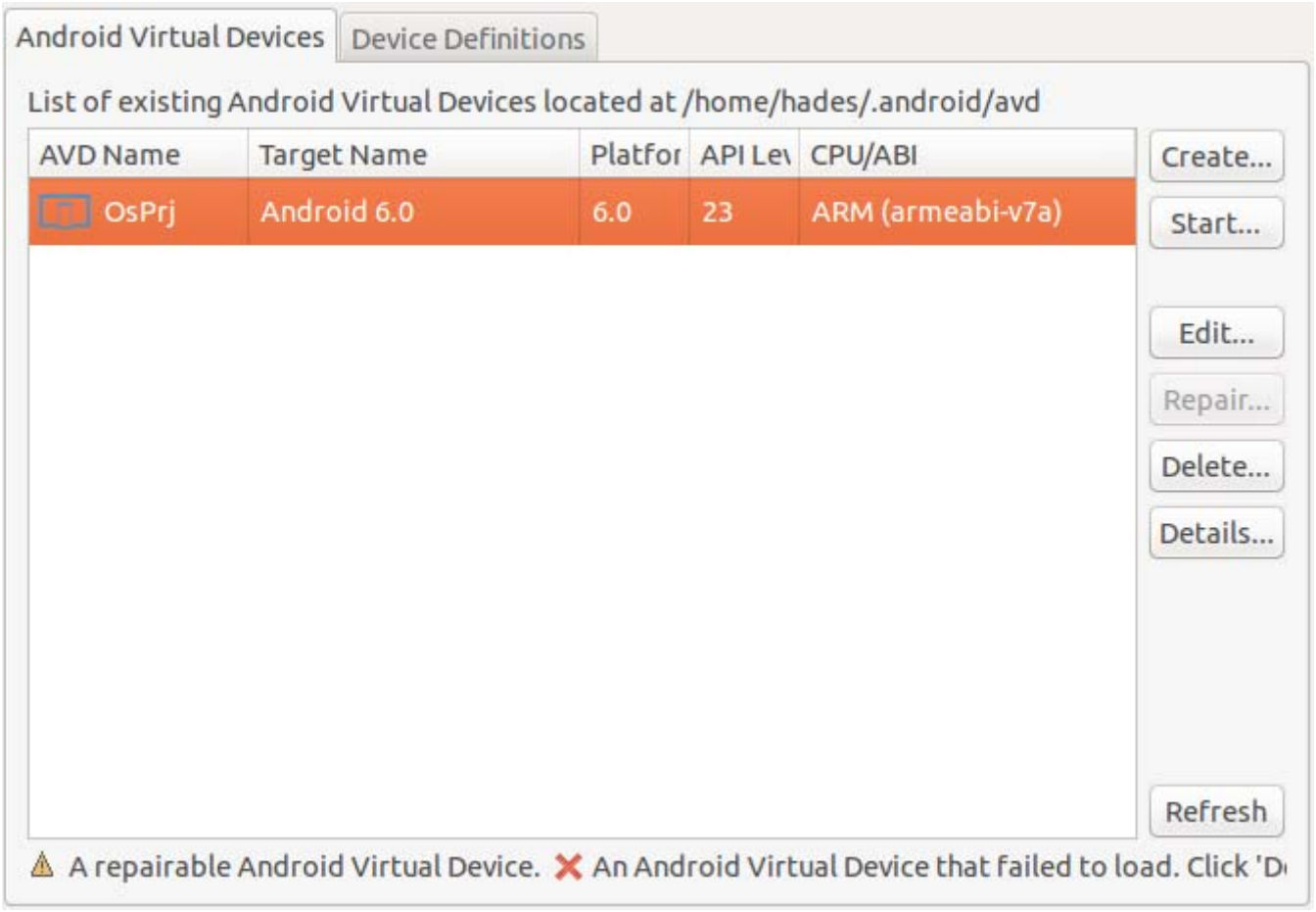
Set Up AVD (cont.)



- You can modify these parameter by yourself except “Target”.
- There will be a warning when RAM is bigger than 768 in Windows.

Set Up AVD (cont.)

- Click Start to start you avd



Set Up AVD (cont.)

- If your Linux is 64-bits, you may get error report when creating avd:
  - Failed to create the SD card.
  - Failed to create sdcard in the AVD fold
- This is because your 64-bits system doesn't have 32-bits lib. Then you should install the necessary lib:
  - sudo apt-get install libc6:i386 libgcc1:i386 gcc-8-base:i386 libstdc++5:i386 libstdc++6:i386
  - or
    sudo apt update && sudo apt install libc6:i386

Set Up AVD (cont.)

- More Error

```
Starting emulator for AVD 'OsPrj'
/home/wangbo/Kit/android-sdk-linux/tools/emulator: error
while loading shared libraries: libstdc++.so.6: cannot open
shared object file: No such file or directory
```

  - sudo apt-get install lib32stdc++6

Set Up NDK

- Because our computer is x86 architecture while most Android devices are ARM architecture, executable files compiled on our computer cannot be executed on the AVD
- We should cross compile the C files using toolchains in NDK

Set Up NDK (cont.)

- For Windows
  - http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-ndk-r11-windows-x86_64.zip
- For Linux
  - http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-ndk-r11-linux-x86_64.zip
- Extract them into a proper location.

Set Up NDK (cont.)

- Extract the NDK files to a proper location.
  - ~/android or /usr/lib/android/ for Linux
  - X:\android-ndk-windows for Windows
- Add location path to your Environment Variables
- Type ndk-build -v to check whether the installation is completed.

An Exception When Setting up AVD

- There is an exception as shown in the figure when setting up AVD, may I ask where is this coming from and how to solve it?
- It appears that some libraries required by Android-SDK is missing in Ubuntu 19.10, which results in the failure of starting AVD. (Earlier Ubuntu versions do not have this problem.)

Please download the zip file from Canvas, check readme.txt, and fix the problem if you are running Ubuntu 19.10.

```
virtual-machine:/usr/local/android-sdk-linux/tools$ ./android avd
Exception in thread "main" java.lang.UnsatisfiedLinkError: no swt-pi-gtk-3550 or swt-pi-gtk in swt.library.path, java.library.path or the jar file
        at org.eclipse.swt.internal.Library.loadLibrary(Unknown Source)
        at org.eclipse.swt.internal.Library.loadLibrary(Unknown Source)
        at org.eclipse.swt.internal.gtk.OS.<clinit>(Unknown Source)
        at org.eclipse.swt.internal.Converter.wcsToMbcs(Unknown Source)
        at org.eclipse.swt.internal.Converter.wcsToMbcs(Unknown Source)
        at org.eclipse.swt.widgets.Display.<clinit>(Unknown Source)
        at com.android.sdkmanager.Main.showAvdManagerWindow(Main.java:427)
        at com.android.sdkmanager.Main.doAction(Main.java:380)
        at com.android.sdkmanager.Main.run(Main.java:151)
        at com.android.sdkmanager.Main.main(Main.java:117)
```

Build Project by NDK

- Make project directory.

  mkdir hello

  mkdir hello/jni

- Put your source code files in JNI folder.

Build Project by NDK (cont.)

- Writing a “Hello World” program
  - For hello.h

```
#ifndef HELLOHEADER_H_
#define HELLOHEADER_H_
#include <stdio.h>
#endif /*HELLOHEADER_H_*/
```

Build Project by NDK (cont.)

- Writing a “Hello World” program
  - For hello.c

```
#include “hello.h”
int main(int argc, char *argv[]){
        printf("Hello World!\n");
        return 0;
}
```

Build Project by NDK (cont.)

- Writing a “Hello World” program
  - For Android.mk which is make file for any project.

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_SRC_FILES := hello.c                  # your source code
LOCAL_MODULE := helloARM                    # output file name
LOCAL_CFLAGS += -pie -fPIE                  # These two line cannot be
LOCAL_LDFLAGS += -pie -fPIE                 # changed.
LOCAL_FORCE_STATIC_EXECUTABLE := true

include $(BUILD_EXECUTABLE)
```

Build Project by NDK (cont.)

- The LOCAL_CFLAGS += -pie -fPIE and LOCAL_LDFLAGS += -pie -fPIE make the program compiled based on PIE. Without these two lines, the program can not be executed in Android.

- Type ndk-build in jni folder

- The executable file is in hello/libs/armeabi

Running on AVD

- To install and run the program you compiled, you can use the multi-purpose Android Debug Bridge (ADB) utility.
- Location of ADB
  - #your sdk location#/platform-tools/
  - You can add this directory to Environment Variables so that you can directly type adb in other directory.

Some ADB command

- To check the AVD status:
  - adb devices
- To move a file to the emulator:
  - adb push #source path ~/hello/helloARM# #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

Running on AVD (cont.)

- After uploading your program file to your AVD, you should type the following command in shell to make it executable:
  - chmod +x #flie name#
  - chmod 777 #flie name#
- Then, you can run your program on AVD.

Linux Modules

- Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand.
- With modules, we can implement some system calls without re-compilation.
- Please study the following example to learn how to use modules.

Modules Source File

- You need to write .c files as the sources to create a module. The following file's name is hello.c.

```c
#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/init.h>
#include<linux/sched.h>
#include<linux/unistd.h>
MODULE_LICENSE("Dual BSD/GPL");
#define __NR_hellocall 356

static int (*oldcall)(void);
static int sys_hellocall(int n, char* str)
{
    printk("this is my system second call!\n the uid = %ld\n str: %s\n",n,str);
    return n;
}
static int addsyscall_init(void)
{
    long *syscall = (long*)0xc000d8c4;
    oldcall = (int(*)(void))(syscall[__NR_hellocall]);
    syscall[__NR_hellocall] = (unsigned long )sys_hellocall;
    printk(KERN_INFO "module load!\n");
    return 0;

}

static void addsyscall_exit(void)
{
    long *syscall = (long*)0xc000d8c4;
    syscall[__NR_hellocall] = (unsigned long )oldcall;
    printk(KERN_INFO "module exit!\n");
}

module_init(addsyscall_init);
module_exit(addsyscall_exit);
```

Modules Source File - Definition

```
#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/init.h>
#include<linux/sched.h>
#include<linux/unistd.h>
MODULE_LICENSE("Dual BSD/GPL");
```

- Properties of module. No need to change them

```
module_init(addsyscall_init);
module_exit(addsyscall_exit);
```

Modules Source File - Functions

```c
static int (*oldcall)(void);
static int addsyscall_init(void)
{
    long *syscall = (long*)0xc000d8c4;
    oldcall = (int(*)(void))(syscall[__NR_hellocall]);
    syscall[__NR_hellocall] = (unsigned long )sys_hellocall;
    printk(KERN_INFO "module load!\n");
    return 0;
}
```

```c
module_init(addsyscall_init);
module_exit(addsyscall_exit);
```

```c
static void addsyscall_exit(void)
{
    long *syscall = (long*)0xc000d8c4;
    syscall[__NR_hellocall] = (unsigned long )oldcall;
    printk(KERN_INFO "module exit!\n");
}
```

Modules Source File – System Call

- You should change this part to accomplish project. Set the syscall number as 356 .

```c
#define __NR_hellocall 356

static int sys_hellocall(int n, char* str)
{
    printk("this is my system second call!\n the uid = %ld\n str: %s\n",n,str);
    return n;
}
```

- Sample of using system call

```c
#include <stdio.h>
int main(){
    printf("This is a test:\n\n");
    int i=syscall(356,123,"test string");
    printf("Answer is %d!\n",i);
    printf("Test End!:\n\n");
    return 0;
}
```

Modules Make File

```
obj-m := hello.o
KID := ~/kernel/goldfish
CROSS_COMPILE=arm-linux-androideabi-
CC=$(CROSS_COMPILE)gcc
LD=$(CROSS_COMPILE)ld

all:
    make -C $(KID) ARCH=arm CROSS_COMPILE=$(CROSS_COMPILE) M=$(shell pwd) modules

clean:
    rm -rf *.ko *.o *.mod.c *.order *.symvers
```

- Save source file and make file in one folder.
- KID is the location of your kernel.
- Add Environment Variable
  - #your ndk location#/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64/bin
- Type make in shell in the folder.
- Then you will get a file *.ko, this is your module.

Use Module

- Upload your .ko file to avd
- Install mod
  - insmod *.ko
- Remove mod
  - rmmod *.ko
- List mod
  - lsmod
- Delete you .ko file before you want to update it.
- Remove the mod installed before you delete .ko file.

Problems

- We have four problems for project 1.
- Problem 1-3 is about implementing a system call with modules.
- Problem 4 is implementing a synchronization algorithm.

Problem 1

- In Linux, we can use ps to check the current processes.
- Furthermore, we can use pstree to see the process tree intuitively.
- In Android, we can use ps, but cannot use pstree

Problem 1

- Write a new system call in Android.
  - The system call you write should take two arguments and return the process tree information in a depth-first-search (DFS) order.
  - Each system call must be assigned a number. Your system call should be assigned number 356.

Problem 1 (cont.)

- 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 */
        pid_t pid;                  /* process id */
        pid_t first_child_pid;      /* pid of youngest child */
        pid_t next_sibling_pid;     /* pid of older sibling */
        long state;                 /* current state of process */
        long uid;                   /* user id of process owner */
        char comm[64];              /* name of program executed */
};
```

You can make some revisions on them if you can get the correct result.

Problem 1 (cont.)

- 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.

- If pointer correlated with the variable in struct prinfo is null, set the value in struct prinfo to 0.
- For example, the first_child_pid should be set to 0 if the process does not have a child.

Problem 1 (cont.)

- Linux maintains a list of all processes in a doubly linked list. Each entry in this list is a task_struct structure, which is defined in include/linux/sched.h. When traversing the process tree data structures, it is necessary to prevent the data structures from changing in order to ensure consistency.
- For this purpose the kernel relies on a special lock, the tasklist_lock. You should grab this lock before you begin the traversal, and only release the lock when the traversal is completed. While holding the lock, your code may not perform any operations that may result in a sleep, such as memory allocation, copying of data into and out from the kernel etc. Use the following code to grab and then release the lock:

```
read_lock(&tasklist_lock);
  ...
  ...
read_unlock(&tasklist_lock);
```

Problem 1 (cont.)

- In order to learn about system calls, you may find it helpful to search the linux kernel for other system calls and see how they are defined. You can use the Linux Cross-Reference(LXR) to investigate different system calls already defined. The files kernel/sched/core.c andkernel/timer.c should provide good reference points for defining your system call.
- You should not try to create your own linked list method for the data structures inside the kernel, but use the existing infrastructure. See include/linux/list.h and look for other places in the kernel where lists are used for examples on how to use them (there are many such places). Also, the course materials contain information about linked lists in the kernel.

Problem 1 (cont.)

- Add system call dynamically.
- Use module.
- But the original android kernel does not support module.
- Compile a New One.
- Kernel is supported on website.
  - http://www.cs.sjtu.edu.cn/~fwu/teaching/res/android-kernel.tar.gz
  - Extract the kernel folder into the user folder.
- Linux Only

Start AVD

- We will start AVD with a new 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.

Some problem

- Apt-get 404 not found.
  - pls try again, the network is not stable.
- AVD is tooooooooo slow.
  - pls be patient.
- Android Debug Bridge (adb) usage

Tips

- task_struct is defined in about line 1270 if you download the Android source code from the website we have provided.
- Some illegal operations (e.g, no-assigned struct pointer) will make your Android virtual device crushed. Be careful.
- Implement the system call with modules. You don't have to revise the kernel code.
- You only need to submit your module's source code for Problem 1.

Problem 2

- Test your new system call
  - Write a simple C program which calls ptree
  - Print the entire process tree (in DFS order) using tabs to indent children with respect to their parents.
  - The output format of every process is:

```
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);
```

Problem 2 – Sample Output

- Example

```
...
init,1,1,0,31,2,0
...
    servicemanager,44,1,1,0,45,1000
    vold,45,1,1,0,47,0
    netd,47,1,1,0,48,0
    debuggerd,48,1,1,0,49,0
    rild,49,1,1,0,50,1001
    surfaceflinger,50,1,1,0,51,1000
    zygote,51,1,1,369,52,0
        system_server,369,1,51,0,421,1000
        ...
        ndroid.launcher,529,1,51,0,550,10008
        ...

...
kthreadd,2,1,0,3,0,0
...
    ksoftirqd/0,3,1,2,0,4,0
    kworker/0:0,4,1,2,0,5,0
    ...
    khelper,6,1,2,0,7,0
    ...
```

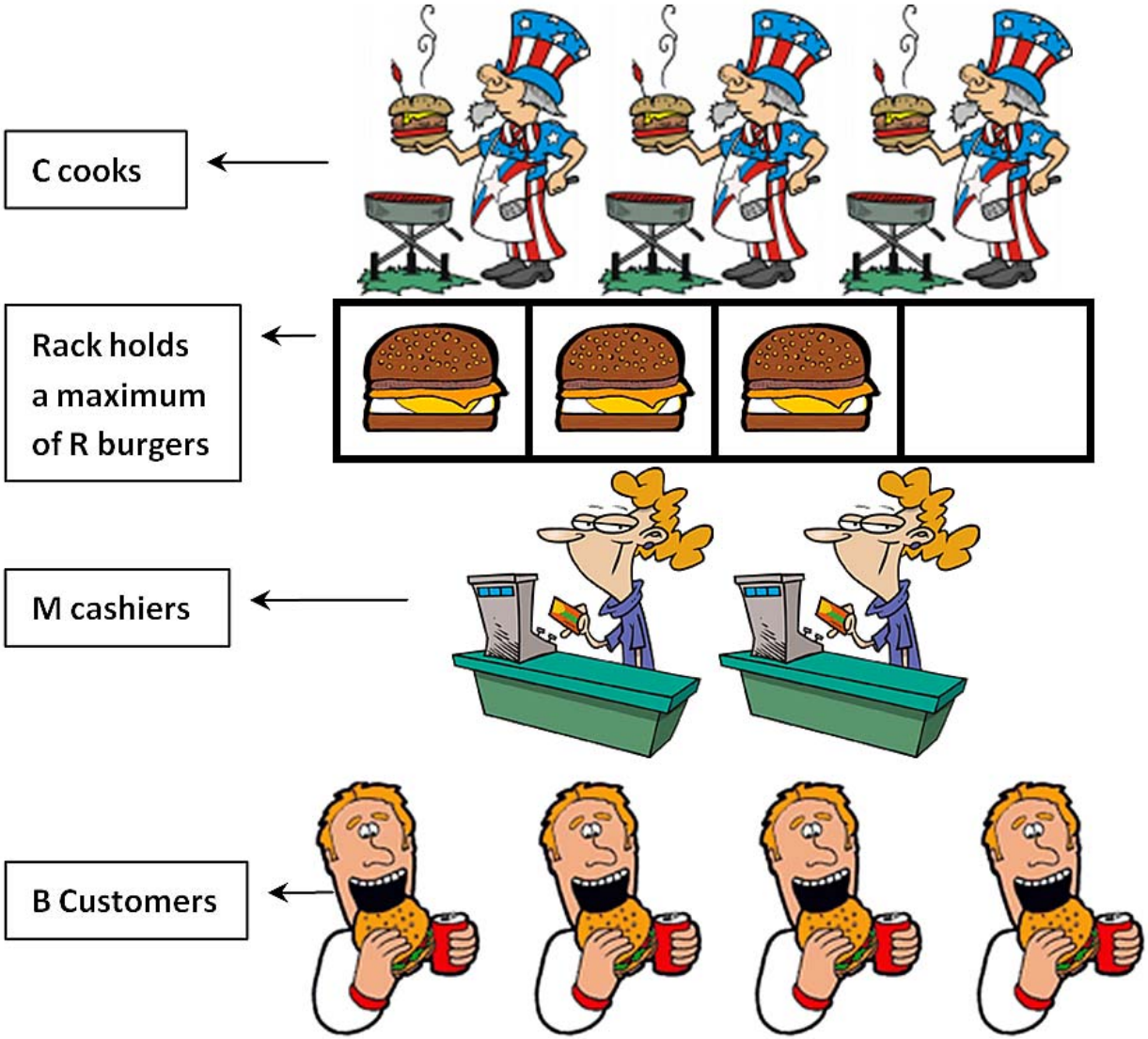
Problem 3

- Generate a new process and output "StudentIDParent" with PID, then generates its child process, which output "StudentIDChild" with PID.
- Use *execl* to execute *ptree* in the child process, show the connection between above two processes.

Problem 4 – Burger Buddies Problem

- Cooks, Cashiers, and Customers are each modeled as a thread.
- Cashiers sleep until a customer is present.
- A Customer approaching a cashier can start the order process.
- A Customer cannot order until the cashier is ready.
- Once the order is placed, a cashier has to get a burger from the rack.
- If a burger is not available, a cashier must wait until one is made.
- The cook will always make burgers and place them on the rack.
- The cook will wait if the rack is full.
- There are NO synchronization constraints for a cashier presenting food to the customer.

Problem 4 – Burger Buddies Problem



Problem 4 – General Requirement

- Source file: BurgerBuddies.c
- Executable file: BBC
- Run: BBC #Cooks #Cashiers #Customers #RackSize

Problem 4 – Sample Output

```
> ./BBC  2  4  41  10
Cooks [2], Cashiers [4], Customers [41]
Begin run.
Cook [1] make a burger.
Cook [1] make a burger.
Cook [2] make a burger.
Customer [10] come.
Casher [3] accepts an order.
Casher [3] take a burger to customor.
Customer [19] come.
Casher [2] accepts an order.
Casher [2] take a burger to customor.
Customer [7] come.
Casher [3] accepts an order.
Casher [3] take a burger to customor.
Customer [17] come.
Casher [2] accepts an order.
Cook [1] make a burger.
Casher [2] take a burger to customor.
……
```

What to Submit

- A“tar” file of your DIRECTORY, containing:
  - “Android.mk”
  - Any “.cc”, “.c”, and “.h” files
  - Any “readme” or “.pdf” files asked for in the project
  - A text file containing the runs of your programs for each of the project parts “testscript”
    - Do not submit ALL runs you have done, just the output required to demonstrate a successful (or unsuccessful) run
    - If you cannot get your program to work, submit a run of whatever you can get to work as you can get partial credit

- DO NOT SUBMIT your object or executable files, remove them before you pack your directory

How to Submit

- Pack your entire Project directory (Only including JNI dircetory)

  tar –cvf Prj1+StudentID.tar project1

- Submit your Prj1+StudentID.tar file on Canvas.

For Help?

- Teaching Assistant
  - Menghan Guo
    - Email: 15667083571@163.com
  - Jiafeng Xu
    - Email: xujiafenga3@163.com

- Some useful website
  - http://www.csdn.net/
  - http://stackoverflow.com/
  - http://developer.android.com/

For Help?

Q&A