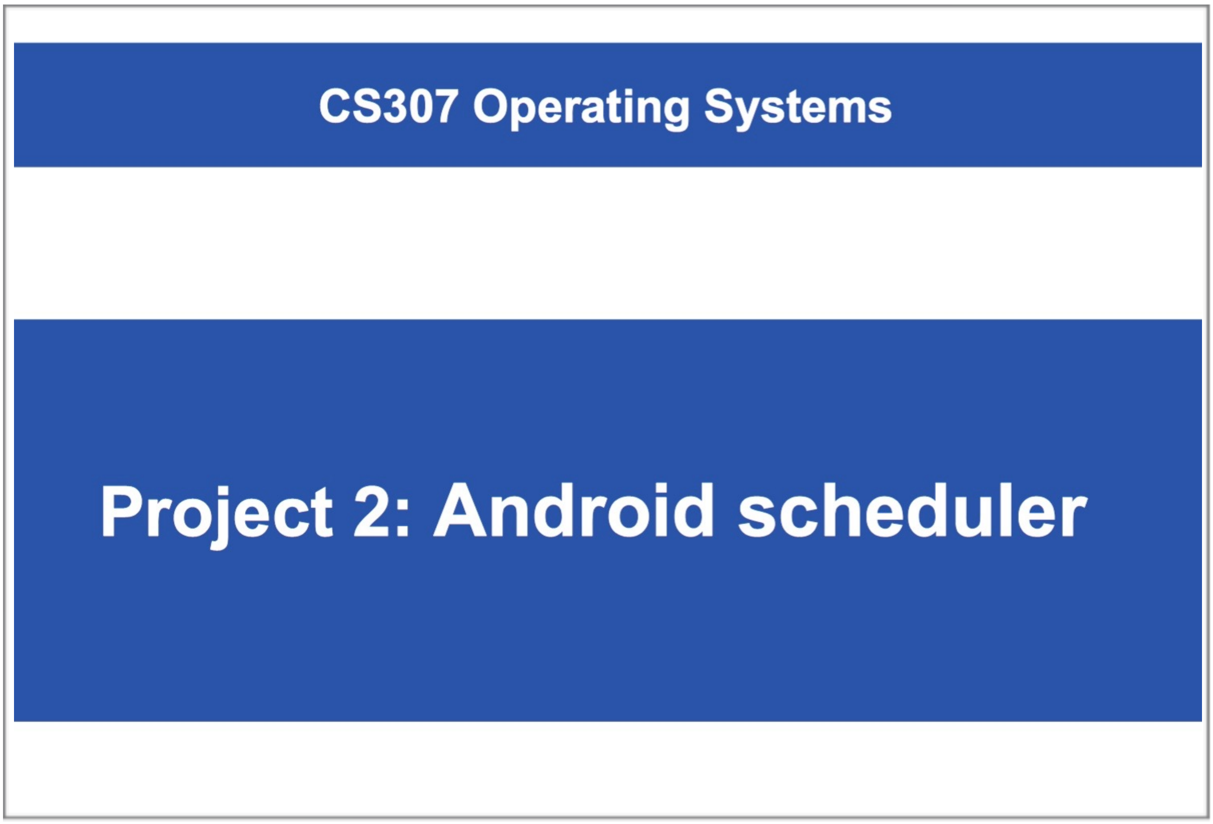
OsPrj2 Report



CS307 Operating Systems

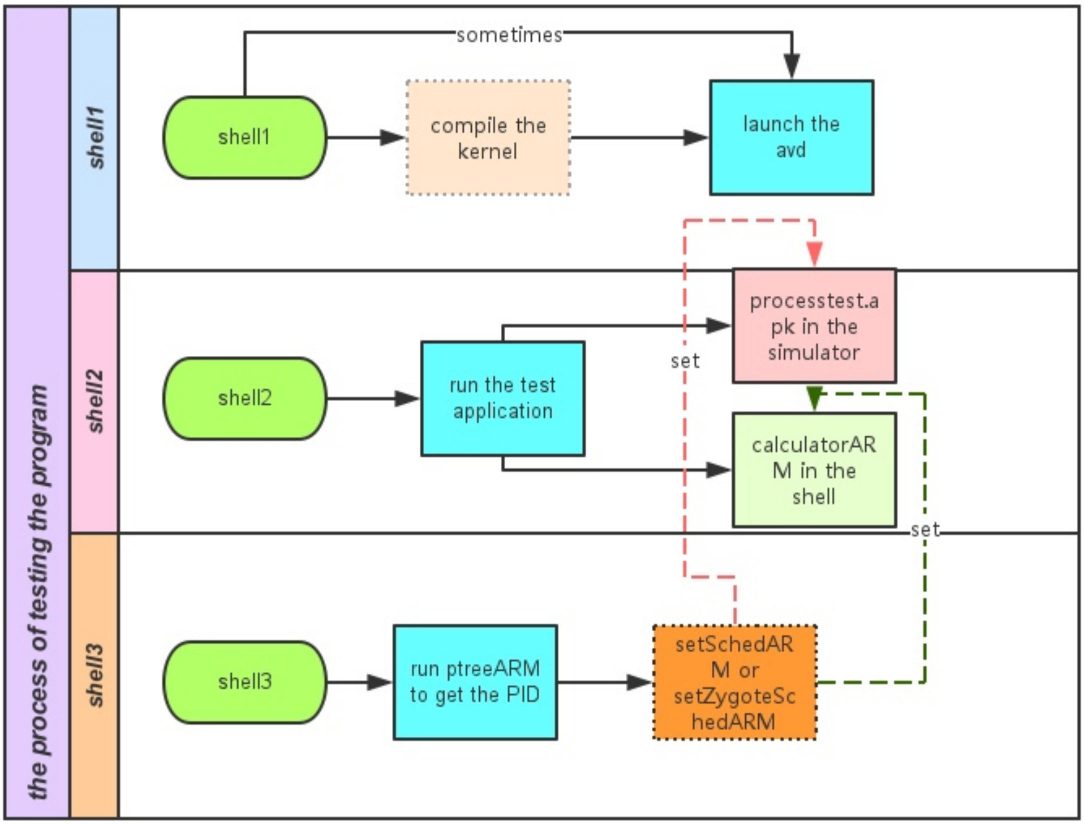
Project 2: Android scheduler

朱舜佳

2016/06/24

Objectives

- Compile the Android kernel
- Familiarize Android scheduler
- Implement a random policy in round robin scheduler
- Get experience with software engineering techniques


the process of testing the program
shell1
shell2
shell3
sometimes
shell1
compile the kernel
launch the avd
processtest.apk in the simulator
set
shell2
run the test application
calculatorARM in the shell
set
shell3
run ptreeARM to get the PID
setSchedARM or setZygoteSchedARM

How I complete the project

The project includes some problems.

1. Problem 1: Compile the Linux kernel

Just following the instructions on the slices, we can compile the kernel without much difficulty.

2. Problem 2: Change the scheduler of test applications

To change the scheduler of a process, the most significant function is sched_setscheduler(), this function can address the process by the pid and set the scheduler and rt_priorty of the process.

Before we run "setSchedARM" in the shell, we first need to run the test application ("processtest" apk in the simulator or "calculatorARM" in the shell), then we run “ptreeARM” or use the command “ps -p” to get the process id of the testing application. After doing these preparations, we can run “setSchedARM” in the shell and input the proper parameters. What we should keep in mind is that the testing application must not be closed during the whole procedure.

The main part of the code is following:

```
//input the parameters
printf("Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): ");
scanf("%d",&myScheduler);
printf("Please input the id of the testprocess : ");
scanf(“%d",&processpid);
printf("Set Process's priority (1-99): “);
scanf("%d", &myPriority);

//set the scheduler and priority
if (sched_setscheduler(processpid, myScheduler, &param) == -1)
{
        perror("sched_setscheduler() failed");
        return -1;
}
```

3. Problem 3 : change the scheduler to all descendants of process zygote to SCHED_RR

The main idea is really similar to the problem 2. The difference is that we should set schedulers for multi processes at one time. This part is following:

```
for (i = i - 1; i >= 0; --i) //exclude for i times
{
        if (sched_setscheduler(processpid[i], myScheduler, &param) == -1)
        {
                perror("sched_setscheduler() failed");
                return -1;
        }
}
```

4. Problem 4: change the default scheduler of all descendants of process zygote.

Problem 4 and 5 change something in the kernel, so we need to compile the kernel after we modify them.

To solve this problem, we must find where the process zygote or main fork its subprocess and change the priority of these subprocesses according to their pids.

At first, I believed I just need to change something in core.c but I failed. Finally I divide my operation into two parts, one is in the core.c and the other is in the fork.c. Both of them are really simple.

In the function sched_fork() in core.c, we add the following code near line 1780:

```
//the subprocess of process main at this point has a comm main;
//the real main process or zygote process is not created in this way.
if(strcmp(p->comm,"main") == 0){
        p->policy = SCHED_RR;
}
```

In the function copy_process() in fork.c, we add the following code near line 178

```
// set the priorty of subprocesses of zygote(main)
if(strcmp(p->real_parent->comm,"main") ==0 )
        p->rt_priority = (p->pid % 5) * 99/ 5 + 1;
```

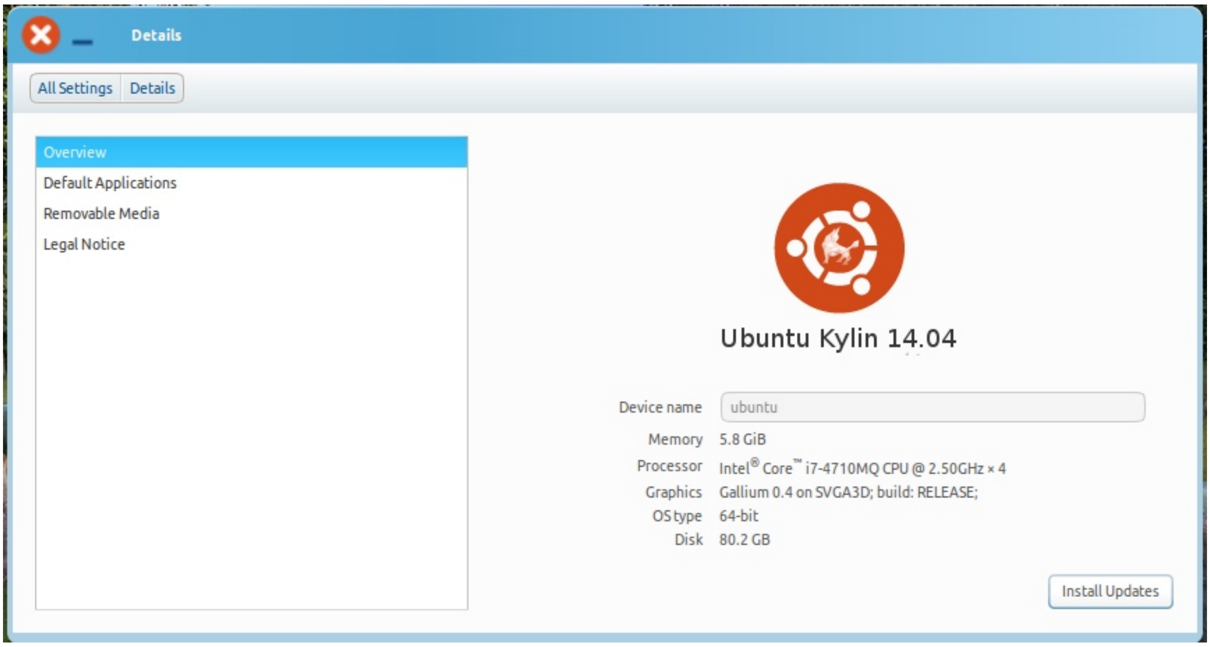
5. Problem 5: change the policy of SCHED_RR to pick the next process randomly.

To address this problem, we need to change something in the rt.c which contains the real time scheduling class of SCHED_RR. Following is the new sched_rt_entity:

```
static struct sched_rt_entity *pick_next_rt_entity(struct rq *rq, struct rt_rq *rt_rq)
{
        struct rt_prio_array *array = &rt_rq->active;
        struct sched_rt_entity *next = NULL;
        struct list_head *queue;
        int idx;

        idx = sched_find_first_bit(array->bitmap);
        BUG_ON(idx >= MAX_RT_PRIO);

        queue = array->queue + idx;  // get the queue of highest priority
        next = list_entry(queue->next, struct sched_rt_entity, run_list); //find the next
sched_rt_entity

        // Here Zhushunjia puts his code
        // get the consistent process by function contain_of()
        struct task_struct *nextprocess  = container_of(next,struct task_struct, rt);
        if(nextprocess->policy != SCHED_RR ) return next; //we do not operate
SCHED_FIFO policy

        struct  list_head *queuehead = queue;

        int random_number;
        int numberofTask = 0;
        get_random_bytes(&random_number,sizeof(random_number)); //get a random
number

        //get the length of the queue.
        while(queue -> next != queuehead) {
         numberofTask++;
         queue = queue->next;
       }

        //get a random number between 0 and numberofTask - 1
        int randomTaskNumber = random_number % numberofTask;
        int i;

        //pick the next sched_rt_entity according to the random number
        queue = queuehead;
        for(i = 0; i < randomTaskNumber; i++ ) {
           queue = queue->next;
           next = list_entry(queue, struct sched_rt_entity, run_list);
           nextprocess  = container_of(next,struct task_struct, rt);
           if(nextprocess->policy != SCHED_RR) return next;
          }
        next = list_entry(queue->next, struct sched_rt_entity, run_list);
        queue = queuehead;

        return next;
}
```

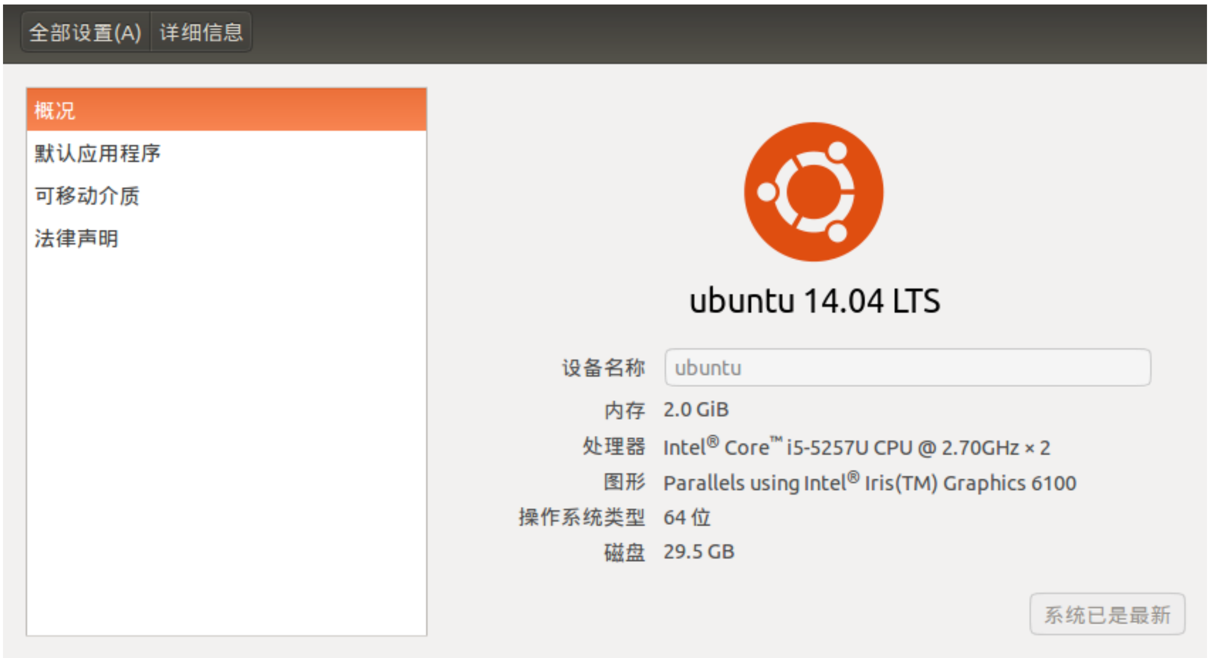
Actual operation results

Configuration of my computers:



Ubuntu Kylin 14.04

Device name ubuntu
Memory 5.8 GiB
Processor Intel® Core™ i7-4710MQ CPU @ 2.50GHz × 4
Graphics Gallium 0.4 on SVGA3D; build: RELEASE;
OS type 64-bit
Disk 80.2 GB

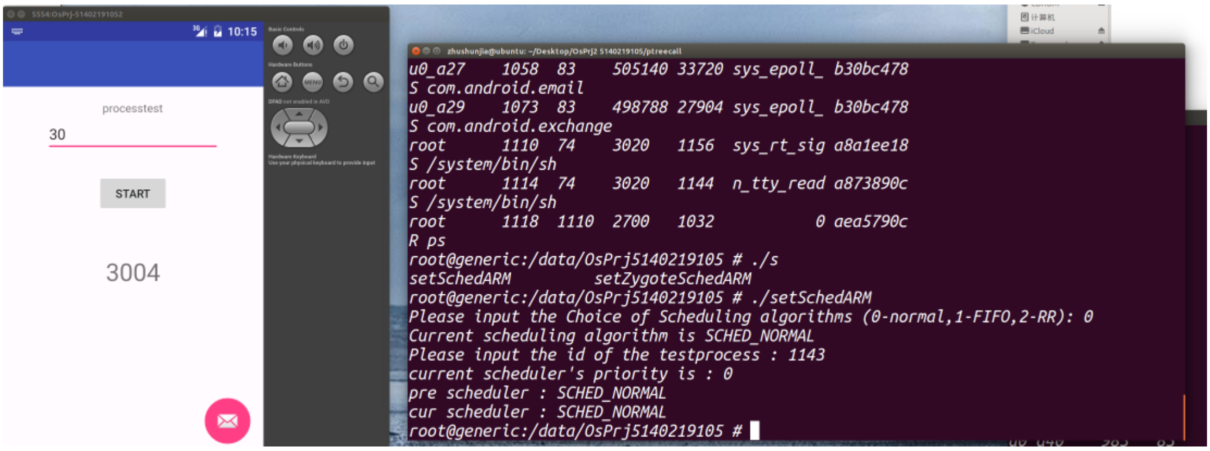


设备名称 ubuntu
内存 2.0 GiB
处理器 Intel® Core™ i5-5257U CPU @ 2.70GHz × 2
图形 Parallels using Intel® Iris(TM) Graphics 6100
操作系统类型 64 位
磁盘 29.5 GB

Processtest.apk:

1. SCHED_NORMAL:

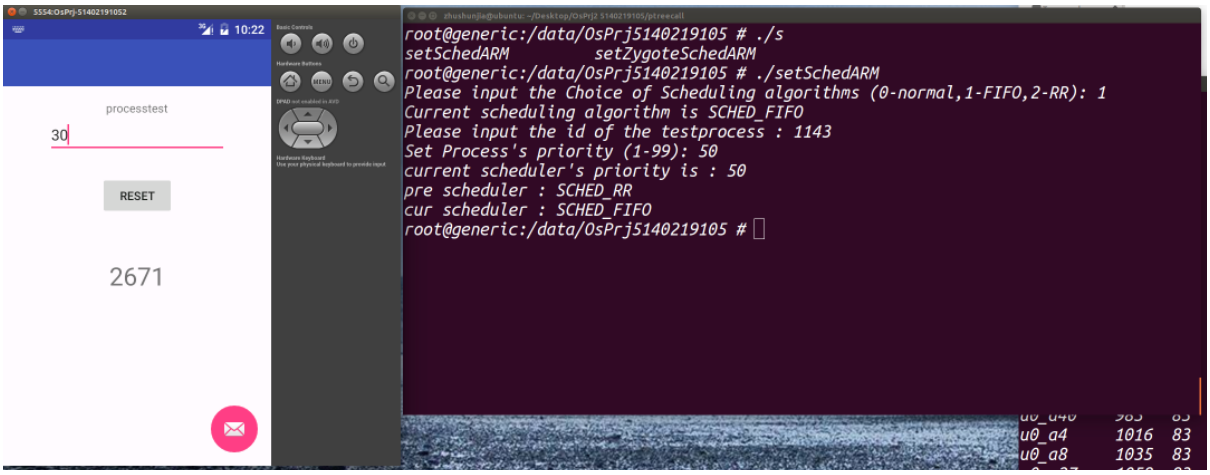
Input: 30 Output: 3004 3219 3103 3133 3007 3152



```
u0_a27    1058  83    505140 33720 sys_epoll_ b30bc478
S com.android.email
u0_a29    1073  83    498788 27904 sys_epoll_ b30bc478
S com.android.exchange
root      1110  74    3020   1156  sys_rt_sig a8a1ee18
S /system/bin/sh
root      1114  74    3020   1144  n_tty_read a873890c
S /system/bin/sh
root      1118  1110  2700   1032           0 aea5790c
R ps
root@generic:/data/OsPrj5140219105 # ./s
setSchedARM          setZygoteSchedARM
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 0
Current scheduling algorithm is SCHED_NORMAL
Please input the id of the testprocess : 1143
current scheduler's priority is : 0
pre scheduler : SCHED_NORMAL
cur scheduler : SCHED_NORMAL
root@generic:/data/OsPrj5140219105 #
```

2. SCHED_FIFO:

Input: 30 Output: 2671 2564 2517 2537 2502 2646

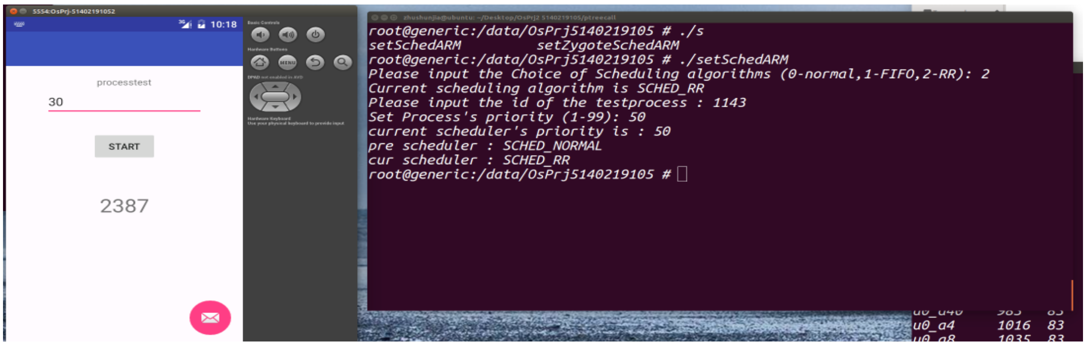


```
root@generic:/data/OsPrj5140219105 # ./s
setSchedARM          setZygoteSchedARM
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 1
Current scheduling algorithm is SCHED_FIFO
Please input the id of the testprocess : 1143
Set Process's priority (1-99): 50
current scheduler's priority is : 50
pre scheduler : SCHED_RR
cur scheduler : SCHED_FIFO
root@generic:/data/OsPrj5140219105 #
```

```
u0_a4      1016   83
u0_a8      1035   83
```

SCHED_RR

Input: 30 Output: 2387 2340 2377 2505 2559 2323

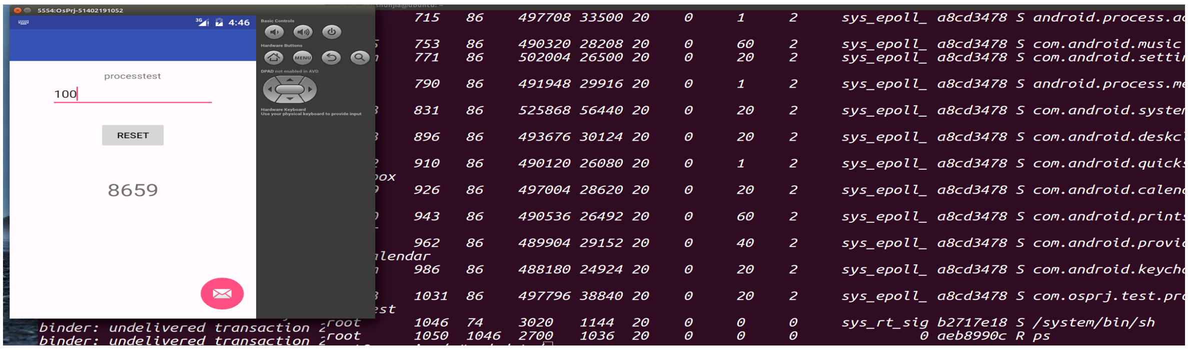


```
root@generic:/data/OsPrj5140219105 # ./s
setSchedARM          setZygoteSchedARM
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 2
Current scheduling algorithm is SCHED_RR
Please input the id of the testprocess : 1143
Set Process's priority (1-99): 50
current scheduler's priority is : 50
pre scheduler : SCHED_NORMAL
cur scheduler : SCHED_RR
root@generic:/data/OsPrj5140219105 #
```

```
u0_a4      1016   83
```

3. Modified SCHED_RR

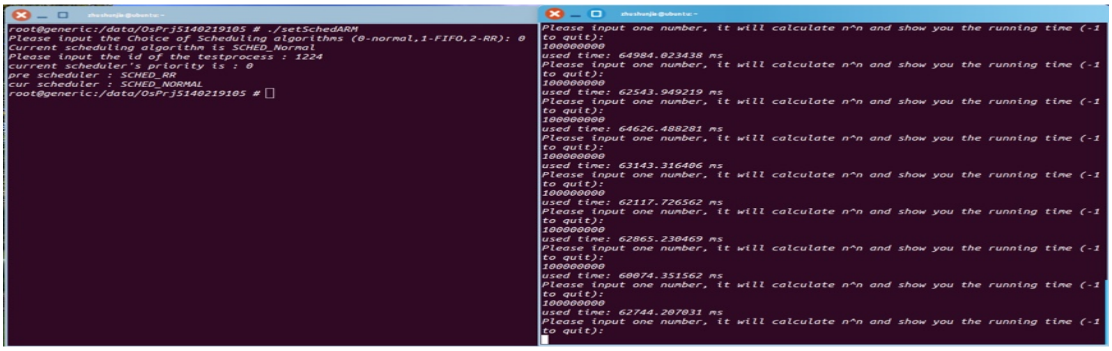
Input: 100 Output:8659 8788 8555



```
   715   86    497708 33500 20    0    1    2     sys_epoll_ a8cd3478 S android.process.ac

   753   86    490320 28208 20    0    60   2     sys_epoll_ a8cd3478 S com.android.music
   771   86    502004 26500 20    0    20   2     sys_epoll_ a8cd3478 S com.android.settin

   790   86    491948 29916 20    0    1    2     sys_epoll_ a8cd3478 S android.process.me

   831   86    525868 56440 20    0    20   2     sys_epoll_ a8cd3478 S com.android.syster

   896   86    493676 30124 20    0    20   2     sys_epoll_ a8cd3478 S com.android.deskc

   910   86    490120 26080 20    0    1    2     sys_epoll_ a8cd3478 S com.android.quicks
box
   926   86    497004 28620 20    0    20   2     sys_epoll_ a8cd3478 S com.android.calenc

   943   86    490536 26492 20    0    60   2     sys_epoll_ a8cd3478 S com.android.prints

   962   86    489904 29152 20    0    40   2     sys_epoll_ a8cd3478 S com.android.provic
lendar
   986   86    488180 24924 20    0    20   2     sys_epoll_ a8cd3478 S com.android.keycho

   1031  86    497796 38840 20    0    20   2     sys_epoll_ a8cd3478 S com.osprj.test.pro
est
binder: undelivered transaction 2root   1046  74    3020  1144  20    0    0    0     sys_rt_sig b2717e18 S /system/bin/sh
binder: undelivered transaction 2root   1050  1046  2700  1036  20    0    0    0              0 aeb8990c R ps
```

calculatorARM:

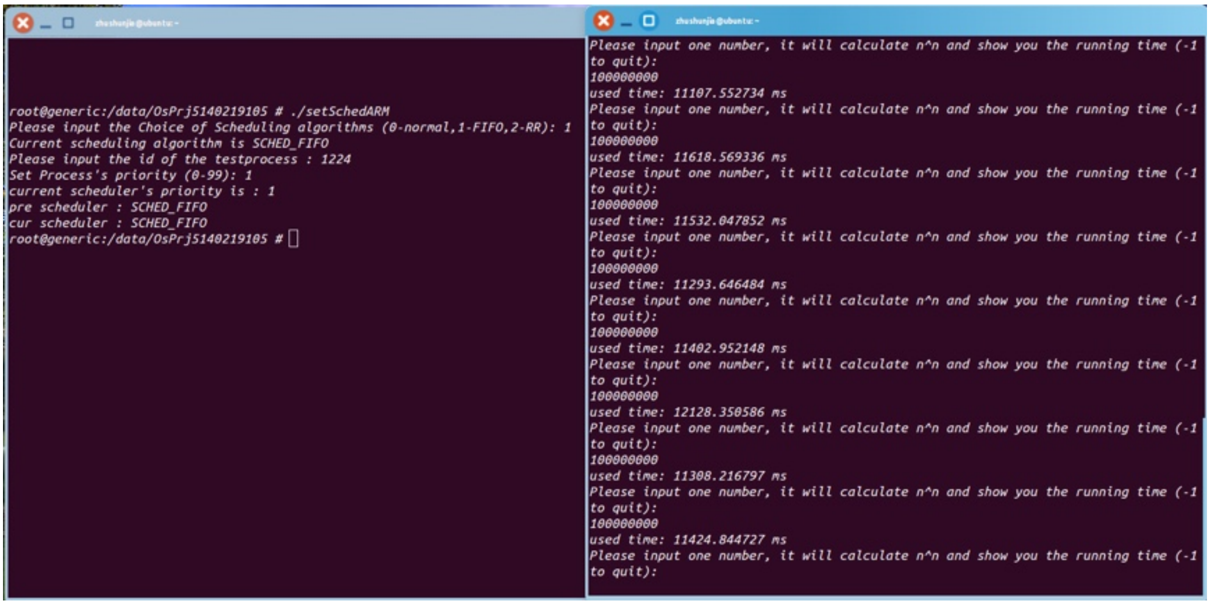
1.SCHED_NORMAL:



```
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 0
Current scheduling algorithm is SCHED_Normal
Please input the id of the testprocess : 1224
current scheduler's priority is : 0
pre scheduler : SCHED_RR
cur scheduler : SCHED_NORMAL
root@generic:/data/OsPrj5140219105 #
```

```
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 64984.023438 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 62543.949219 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 64626.488281 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 63143.316406 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 62117.726562 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 62865.230469 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 60074.351562 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 62744.207031 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
```

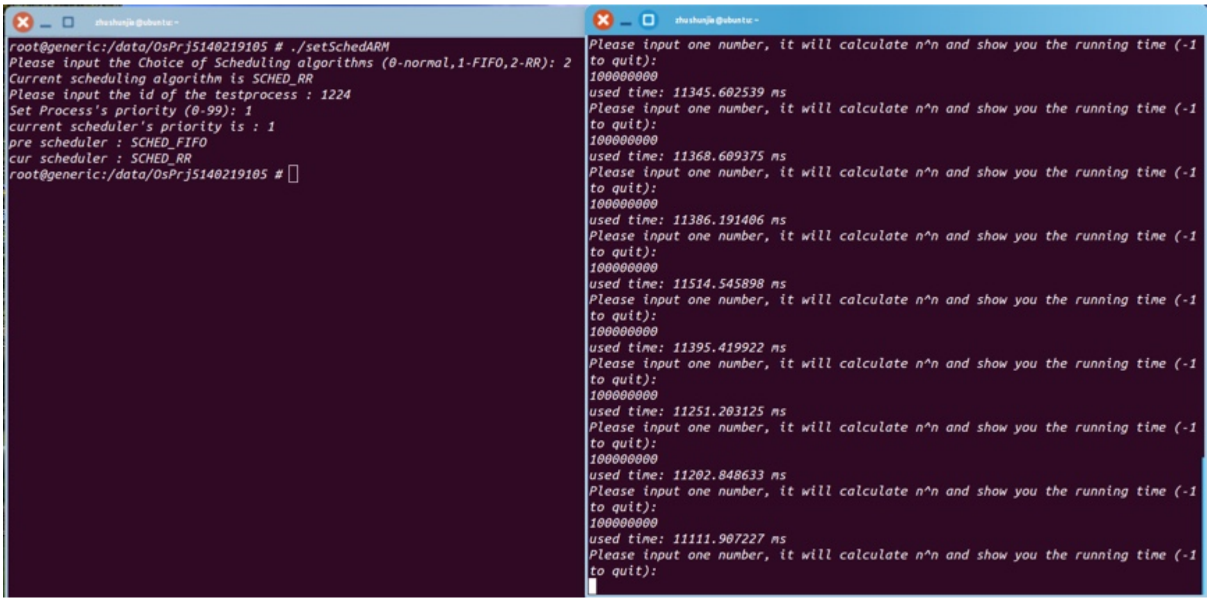
2.SCHED_FIFO:



```
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 1
Current scheduling algorithm is SCHED_FIFO
Please input the id of the testprocess : 1224
Set Process's priority (0-99): 1
current scheduler's priority is : 1
pre scheduler : SCHED_FIFO
cur scheduler : SCHED_FIFO
root@generic:/data/OsPrj5140219105 #
```

```
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11107.552734 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11618.569336 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11532.047852 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11293.646484 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11402.952148 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 12128.350586 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11308.216797 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11424.844727 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
```

3.SCHED_RR：



```
root@generic:/data/OsPrj5140219105 # ./setSchedARM
Please input the Choice of Scheduling algorithms (0-normal,1-FIFO,2-RR): 2
Current scheduling algorithm is SCHED_RR
Please input the id of the testprocess : 1224
Set Process's priority (0-99): 1
current scheduler's priority is : 1
pre scheduler : SCHED_FIFO
cur scheduler : SCHED_RR
root@generic:/data/OsPrj5140219105 #
```

```
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11345.602539 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11368.609375 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11386.191406 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11514.545898 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11395.419922 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11251.203125 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11202.848633 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
100000000
used time: 11111.907227 ms
Please input one number, it will calculate n^n and show you the running time (-1
to quit):
```

Compare three Scheduling policy and Analysis of the result：

SCHED_NORMAL

This is how most normal applications are run. The amount of cpu each process consumes and the latency it will get is mostly determined by the 'nice' value. They run for short periods and share cpu amongst all other processes running with the same policy, across all nice values.

SCHED_FIFO

The processes schedule according to their realtime priority which is unrelated to the nice value. The highest priority process runs indefinitely, never releasing the cpu except to an even higher priority realtime task or voluntarily. Only proper realtime code should ever use this policy as the potential for hardlocking a machine is high if the process runs away. Audio applications for professional performance such as jack use this policy.

SCHED_RR

The processes run similar to SCHED_FIFO except that if more than one process has the same realtime priority, they will run for short periods each and share the cpu.

According to the result above, we can draw the conclusion that the same processes under SCHED_FIFO or SCHED_RR will run faster than that whose scheduler policy is SCHED_NORMAL, and SCHED_FIFO and SCHED_RR have the similar performance. This is because these two scheduling policies are applied by real time processes which have higher priority than the common ones. The real time processes can be operated in a limited time.

There is a confusing phenomenon that “processtest.apk” will have a better performance when the input number is comparatively large (over 50). May this is partly due to the fact we run the application on a simulator.

The origin SCHED_RR is sensible to the priority of the process. Consider the situation that there are many real time processes that run with SCHED_RR policy and they all have higher priority than the test process. In this case, the test process will not occupy the CPU immediately. It usually need to wait for the processes of higher priority. The modified SCHE_RR is less related to the priority of the process because it just pick the next process at random.

Thoughts and Feelings

I do consider this project as a meaningful one and it helps me better understand some scheduling algorithms, the Linux kernel and the android system.

To complete this hard task, I refer to many resources on the Internet. I have learnt a lot from those internet forums like Stackoverflow and CSDN. Of course, I also discussed these problems I faced with other students. And I think that we all benefit a lot during the procedure of working on the project.

I encountered many problems when I want to change the kernel. The virtual machine crashed down for several times ，Thankfully, I used the app TimeMachine to recover my system and it did work! It is difficult for me to understand the whole kernel system of Linux. However, I am still attracted by its elegant and brief principle.

Finally, thanks for TA Wang 's kindness and useful suggestions!