

## Project 2: Android Scheduler

Fan Wu

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

# Objectives

---

1. Compile the Android kernel.
2. Familiarize Android scheduler
3. Implement a *weighted round robin* scheduler.
4. Get experience with software engineering techniques.

# Preliminary

# Enviroment

## ■ Implementation

- AVD(Android Virtual Devices)
  - ▶ SDK version r24.4.1

## ■ Development

- Linux (64-bits)
  - ▶ Ubuntu (recommended)
  - ▶ Debian
  - ▶ Fedora
- VMware

# What to Submit

- A “tar” file of your DIRECTORY, containing:
  - All \*.c, \*.h files you have changed in Linux kernel.
  - Any “readme” or “.pdf” files asked for in the project
  - Screen captures of the scheduler test
    - ▶ 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 code in a project directory  
`tar -cvf Prj2+StudentID.tar project1`
- Send your `Prj2+StudentID.tar` file on Canvas.

# For Help?

---

## ■ Teaching Assistant

- Menghan Guo

- ▶ Email: 15667083571@163.com

- Jiafeng Xu

- ▶ Email: xujiafenga3@163.com

# Recompile the Kernel



# Compile the Linux Kernel

- Make sure that your environment variables are correct.

```
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
export PATH=~/Kit/android-sdk-linux/platform-tools:$PATH
export PATH=~/Kit/android-sdk-linux/tools:$PATH
export PATH=~/Kit/android-ndk-linux:$PATH
export PATH=~/Kit/android-ndk-linux/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64/bin:$PATH
```

# Compile the Linux Kernel (cont.)

## ■ Modify Makefile in the kernel

### ● Change

- ▶ ARCH                      ?=       \$(SUBARCH)
- ▶ CROSS\_COMPILE        ?=

### ● To

```
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH                ?= arm
CROSS_COMPILE       ?= arm-linux-androideabi-
# Architecture as present in compile.h
```

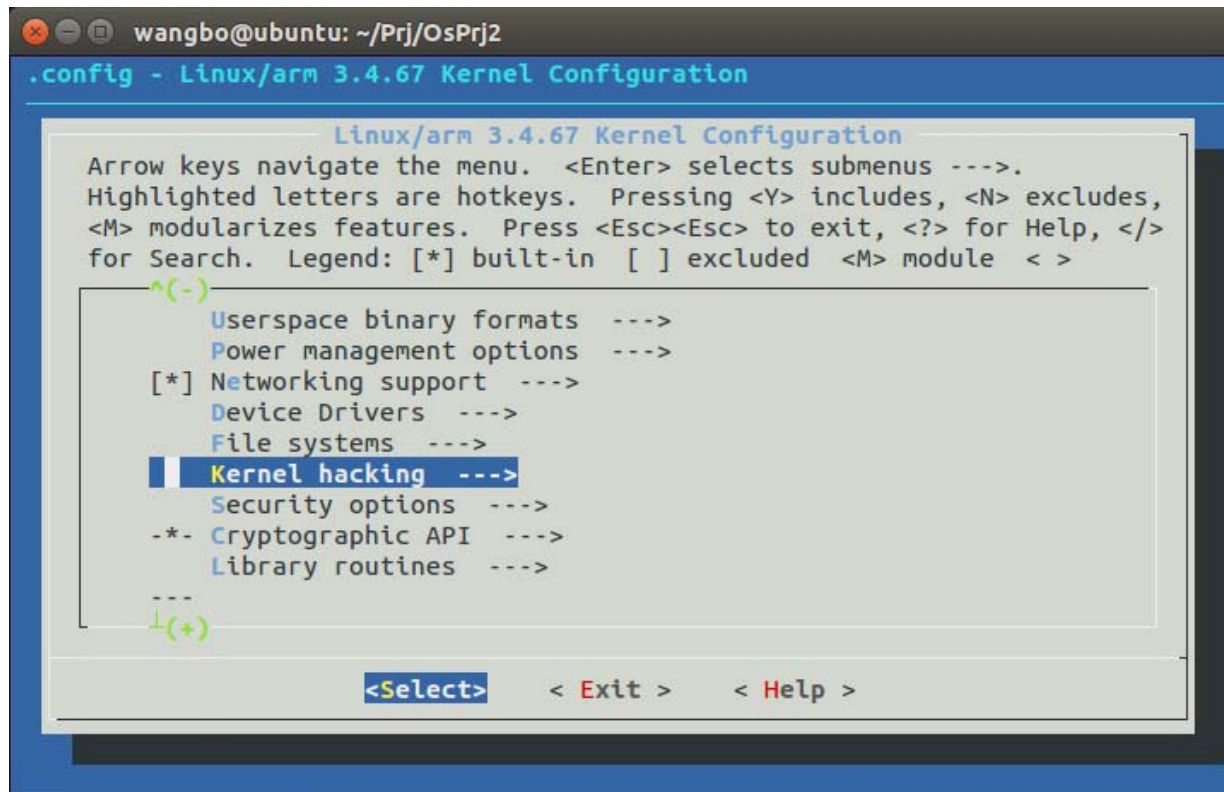
# Compile the Linux Kernel (cont.)

- Execute the following command:

```
wangbo@ubuntu:~/Prj/0sPrj2$ make goldfish_armv7_defconfig
#
# configuration written to .config
#
wangbo@ubuntu:~/Prj/0sPrj2$ sudo apt-get install ncurses-dev
wangbo@ubuntu:~/Prj/0sPrj2$ make menuconfig
```

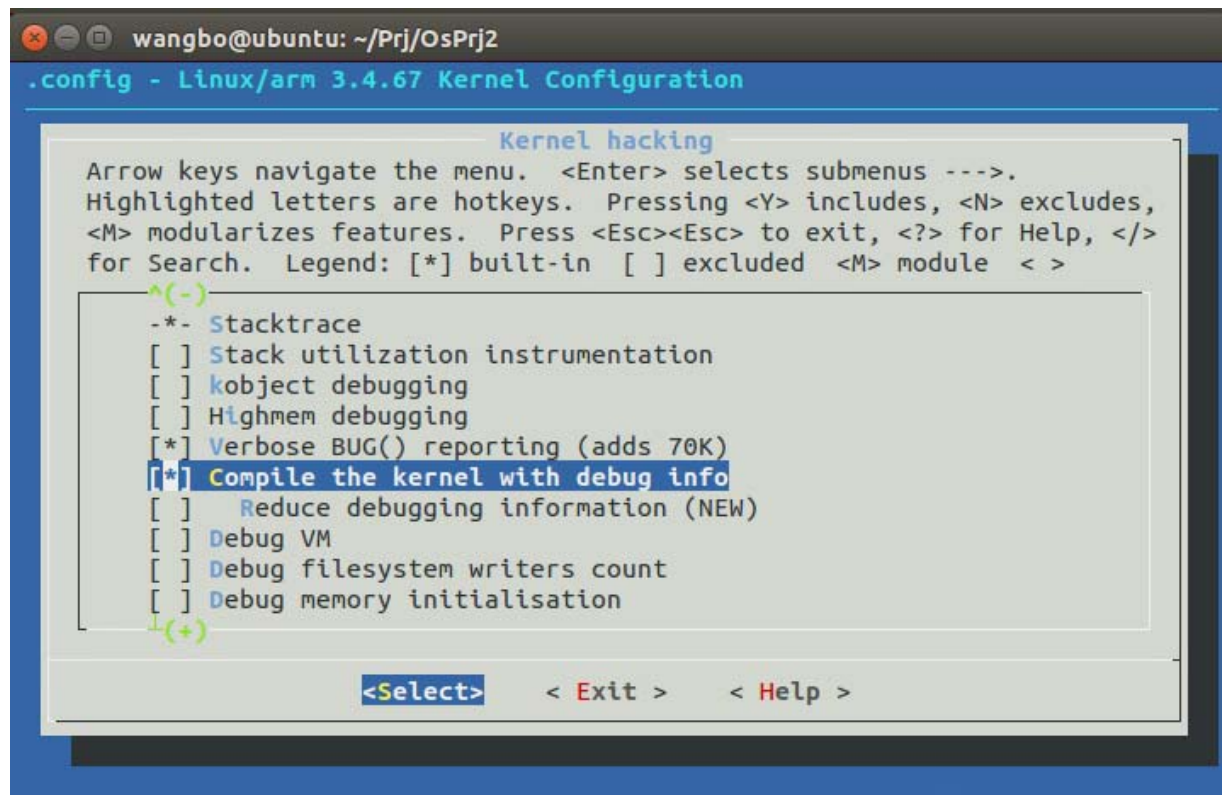
# Compile the Linux Kernel (cont.)

- Then you can see a GUI configuration dialog:



# Compile the Linux Kernel (cont.)

- Open the *Compile the kernel with debug info* in *Kernel hacking*:



```
wangbo@ubuntu: ~/Prj/OsPrj2
.config - Linux/arm 3.4.67 Kernel Configuration

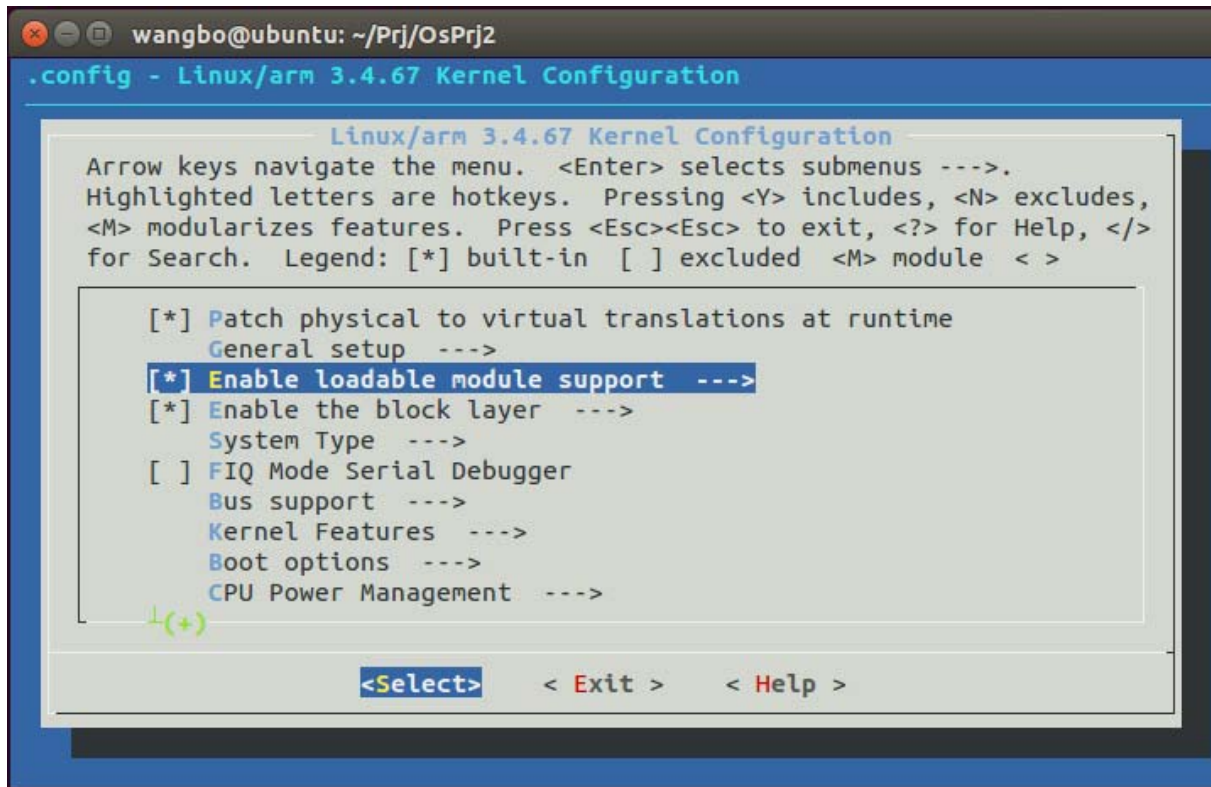
Kernel hacking
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

^(-)
-* Stacktrace
[ ] Stack utilization instrumentation
[ ] kobject debugging
[ ] Highmem debugging
[*] Verbose BUG() reporting (adds 70K)
[*] Compile the kernel with debug info
[ ] Reduce debugging information (NEW)
[ ] Debug VM
[ ] Debug filesystem writers count
[ ] Debug memory initialisation
(+)
```

<Select> < Exit > < Help >

# Compile the Linux Kernel (cont.)

- *Enable loadable module support* with Forced module loading, Module unloading and Forced module unloading in it:



```
wangbo@ubuntu: ~/Prj/OsPrj2
.config - Linux/arm 3.4.67 Kernel Configuration

Linux/arm 3.4.67 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

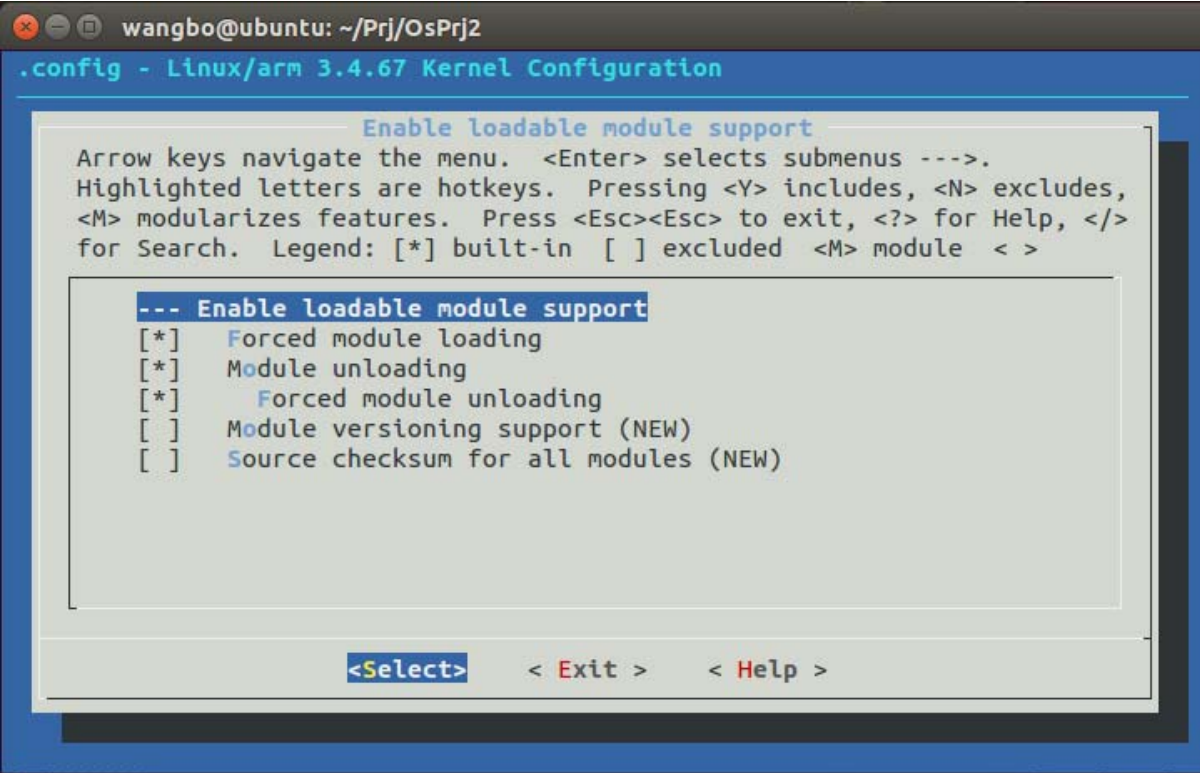
[*] Patch physical to virtual translations at runtime
    General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    System Type --->
[ ] FIQ Mode Serial Debugger
    Bus support --->
    Kernel Features --->
    Boot options --->
    CPU Power Management --->

+(<+>)

<Select>  < Exit >  < Help >
```

# Compile the Linux Kernel (cont.)

- *Enable loadable module support* with Forced module loading, Module unloading and Forced module unloading in it:



```
wangbo@ubuntu: ~/Prj/OsPrj2
.config - Linux/arm 3.4.67 Kernel Configuration

Enable loadable module support
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

--- Enable loadable module support
[*] Forced module loading
[*] Module unloading
[*] Forced module unloading
[ ] Module versioning support (NEW)
[ ] Source checksum for all modules (NEW)

<Select> < Exit > < Help >
```

# Compile the Linux Kernel (cont.)

## ■ Compile it

- The number of -j\* depends on the number of cores of your system.

```
wangbo@ubuntu:~/Prj/OsPrj2$ make -j4
SYSMAP    System.map
SYSMAP    .tmp_System.map
OBJCOPY   arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
GZIP      arch/arm/boot/compressed/piggy.gzip
AS        arch/arm/boot/compressed/piggy.gzip.o
LD        arch/arm/boot/compressed/vmlinux
OBJCOPY   arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
wangbo@ubuntu:~/Prj/OsPrj2$
```



# Introduction to Project 2

# Problem

- Android supports three built-in scheduling policies: Normal, FIFO and RR.
- In this problem, you are required to implement a new scheduling policy:
  - **Weighted Round Robin.**
- The blog relevant to *linux kernel scheduler* could be helpful to you:
  - <https://helix979.github.io/jkoo/post/os-scheduler/>

# Problem

- **Weighted Round Robin (WRR)**
  - Round-robin scheduling treats all tasks equally, but there are times when it is desirable to give some tasks preference over the others.
  - Android Tasks can be classified into **foreground groups** and **background groups**.
  - WRR assigned more milliseconds as a time slice for **foreground groups**. (In our problem, 100ms for fore and 10ms for back)

# Problem

## ■ Foreground and Background Groups

- From user level, you can run `ps -P` on the device or emulator to check the assigned groups for each task.
- At the kernel level, a task's *group information* can be found using a task pointer. Refer to the line 96 in `kernel/sched/debug.c` and use that function appropriately.
- The return value will be `"/` for a foreground (and system group in earlier versions) group, `"/bg_non_interactive` for a background group.

# Implementation Details

- Files **MAY NEED** Modification
  - /arch/arm/configs/goldfish\_armv7\_defconfig
  - /include/linux/sched.h
  - /kernel/sched/core.c
  - /kernel/sched/sched.h
- To implement WRR scheduler, you need to create a new class in the directory /kernel/sched/, that is
  - **/kernel/sched/wrr.c**

**IMPORTANT:** If you feel confused on what to do in wrr.c, read /kernel/sched/rt.c carefully. In rt.c, RR and FIFO are well implemented.

# Implementation Details

/arch/arm/configs/goldfish\_armv7\_defconfig

- In this file, you need to add a new line as follows:

```
13 CONFIG_WRR_GROUP_SCHED=y
14 CONFIG_RT_GROUP_SCHED=y
15 CONFIG_BLK_DEV_INITRD=y
```

# Implementation Details

## /include/linux/sched.h

In this file, you need to:

- Define **SCHED\_WRR** (Refer to SCHED\_RR, about Line 42). The value of SCHED\_WRR should be **6**.
- Define **sched\_wrr\_entity** (Refer to sched\_rt\_entity, about Line 1250)
- Define **time slice** for foreground and background groups. (Refer to RR\_TIMESLICE, about Line 1280)
- Add a **sched\_wrr\_entity** variable to task\_struct (About Line 1310).
- Declare a **wrr\_rq** struct. (Refer to struct cfs\_rq, about Line 150)
- **Maybe a little more to be revised**. It depends on your implementation.

# Implementation Details

**/kernel/sched/sched.h** (path is different from last page)

In this file, you need to:

- Declare a **wrr\_rq** struct. (Refer to struct **rt\_rq**, about Line 90)
- Define a new struct **wrr\_rq** (Refer to **rt\_rq**, about Line 300).
- Add a **wrr\_rq** variable to struct **rq** (About line 400) and similarly add a **list\_head** variable as the figure shows.

```
415 #ifdef CONFIG_RT_GROUP_SCHED
416     struct list_head leaf_rt_rq_list;
417 #endif
418 #ifdef CONFIG_WRR_GROUP_SCHED
419     struct list_head leaf_wrr_rq_list;
420 #endif
421
```

- Declare some **extern variables and functions (\*)**. (You can refer to extern var/func of **rt** in the same file, About Line 190-210, Line 880, Line 900, Line 1170-1180). E.g.

```
883 extern const struct sched_class stop_sched_class;
884 extern const struct sched_class rt_sched_class;
885 extern const struct sched_class fair_sched_class;
886 extern const struct sched_class idle_sched_class;
887 extern const struct sched_class wrr_sched_class;
```

- **Maybe a little more to be revised.** It depends on your implementation.



# Implementation Details

/kernel/sched/core.c

In this file, you need to:

- Revise function: `static void __sched_fork(struct task_struct *p)`
- Revise function: `static void __setscheduler(struct rq *rq, struct task_struct *p, int policy, int prio)`
- Revise function: `static void __sched_setscheduler(struct task_struct *p, int policy, const struct sched_param *param, bool user)`
- Add `init_wrr_rq(&rq->wrr)`. (Refer to `init_rt_rq()`, about Line 7230)
- Revise function: `static void free_sched_group(struct task_group *tg)` and `struct task_group *sched_create_group(struct task_group *parent)` (about Line 7500-7600)
- **Maybe a little more to be revised.** It depends on your implementation.

# Implementation Details

/kernel/sched/core.c

- To know what to revise, you have to read the code carefully and know what are they implemented for.
- For example, when revising `__sched_setscheduler`, we meet the following code segment:

```
1280
1281     if (policy != SCHED_FIFO && policy != SCHED_RR &&
1282         policy != SCHED_NORMAL && policy != SCHED_BATCH &&
1283         policy != SCHED_IDLE)
```

Since we have one more policy WRR now, we should change it to:

```
281     if (policy != SCHED_FIFO && policy != SCHED_RR &&
282         policy != SCHED_NORMAL && policy != SCHED_BATCH &&
283         policy != SCHED_IDLE && policy != SCHED_WRR)
284         return -EINVAL;
```

# Implementation Details

## /kernel/sched/wrr.c

- This is the *major file* in which you write codes. You can refer to rt.c in the same directory to learn how to write wrr.c
- Here, we give a framework of wrr\_sched\_class

```
const struct sched_class wrr_sched_class = {  
    .next = &fair_sched_class, /*Required*/  
    .enqueue_task = enqueue_task_wrr, /*Required*/  
    .dequeue_task = dequeue_task_wrr, /*Required*/  
    .yield_task = yield_task_wrr, /*Required*/  
    .check_preempt_curr = check_preempt_curr_wrr, /*Required*/  
  
    .pick_next_task = pick_next_task_wrr, /*Required*/  
    .put_prev_task = put_prev_task_wrr, /*Required*/  
  
    .task_fork = task_fork_wrr,  
#ifdef CONFIG_SMP  
    .select_task_rq = select_task_rq_wrr, /*Never need impl*/  
    .set_cpus_allowed = set_cpus_allowed_wrr,  
    .rq_online = rq_online_wrr, /*Never need impl*/  
    .rq_offline = rq_offline_wrr, /*Never need impl*/  
    .pre_schedule = pre_schedule_wrr, /*Never need impl*/  
    .post_schedule = post_schedule_wrr, /*Never need impl*/  
    .task_woken = task_woken_wrr, /*Never need impl*/  
#endif  
    .switched_from = switched_from_wrr, /*Never need impl*/  
  
    .set_curr_task = set_curr_task_wrr, /*Required*/  
    .task_tick = task_tick_wrr, /*Required*/  
  
    .get_rr_interval = get_rr_interval_wrr,  
  
    .prio_changed = prio_changed_wrr, /*Never need impl*/  
    .switched_to = switched_to_wrr, /*Never need impl*/  
};
```

# Implementation Details

/kernel/sched/wrr.c

For functions labeled “Required”, you need to implement it in wrr.c

```
static void switched_to_wrr(struct rq *rq, struct task_struct *p)
{
-- 14 lines: struct sched_wrr_entity *wrr_entity = &p->wrr;
}
```

For functions labeled “Never need impl”, you can just put them dummy

```
static void prio_changed_wrr(struct rq *rq, struct task_struct *p,
                             int oldprio)
{
}
```

# Implementation Details

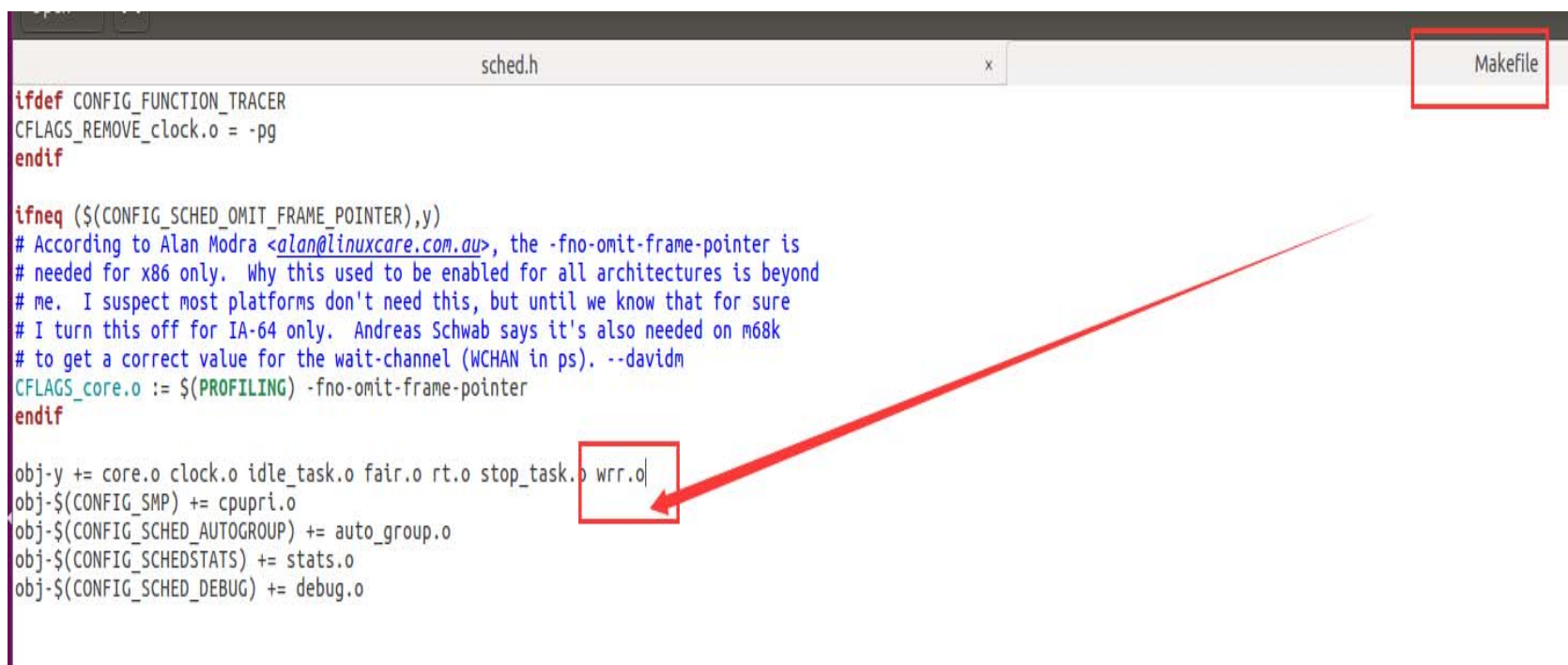
## /kernel/sched/wrr.c

- Remember that in wrr.c, when you want to allocate time slice to a task with WRR as its policy, you should judge whether it is a foreground task or background task, and allocate corresponding time slice to it.
- The blog relevant to *linux kernel scheduler* could be helpful to you:
  - <https://helix979.github.io/jkoo/post/os-scheduler/>

# Implementation Details

To put your */kernel/sched/wrr.c* into effect:

- You need to revise the **Makefile** in */kernel/sched* like this:



```

sched.h
x
Makefile

ifdef CONFIG_FUNCTION_TRACER
CFLAGS_REMOVE_clock.o = -pg
endif

ifneq ($(CONFIG_SCHED_OMIT_FRAME_POINTER),y)
# According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
# needed for x86 only. Why this used to be enabled for all architectures is beyond
# me. I suspect most platforms don't need this, but until we know that for sure
# I turn this off for IA-64 only. Andreas Schwab says it's also needed on m68k
# to get a correct value for the wait-channel (WCHAN in ps). --davidm
CFLAGS_core.o := $(PROFILING) -fno-omit-frame-pointer
endif

obj-y += core.o clock.o idle_task.o fair.o rt.o stop_task.o wrr.o
obj-$(CONFIG_SMP) += cpupri.o
obj-$(CONFIG_SCHED_AUTOGROUP) += auto_group.o
obj-$(CONFIG_SCHEDSTATS) += stats.o
obj-$(CONFIG_SCHED_DEBUG) += debug.o
```

For Your Work

# What to show

## Basic

- We will provide an apk *processtest.apk*.
- Add some `printf()` in *wrr.c* or some other places which proves there is a task using WRR as a policy.
- Write a **test file**, which can change the scheduler in user space.
  - Change the apk's scheduler to WRR when the apk is in foreground groups, and give out some information (pid, name, timeslice, and some others you like).
  - Change the apk's scheduler to WRR when the apk is in background groups, and give out some information (pid, name, timeslice, and some others you like).
- To check the pid of *processtest.apk* in Android shell, you may use:

```
root@generic:/ # ps -P|grep processtest
u0_a53 1191 86 574196 47772 fg sys_epoll_abb59478 S com.osprj.test.processtest
```



# What to show

## Basic

Print information when app is in foreground:

```
Please input the Choice of Scheduling algorithms (0-NORMAL,1-FIFO,2-RR,6-WRR): 6
Current scheduling algorithm is SCHED_WRR
Please input the id of the testprocess : 1225
Set Process's priority (1-99): 60
current scheduler's priority is : 60
pre scheduler : SCHED_NORMAL
cur scheduler : SCHED_WRR
```

Print information when app is in background:

```
I AM IN __SCHED_SETSCHEDULER
group=/bg_non_interactive
Switched to a backgroup WRR entity, pid=1084, proc=est.proce
sstest,
```

# What to show

---

## Bonus (10 points in Final Score):

- Any extended ideas can be considered into the bonus!
- Here are some of the ideas we provide, I hope you won't be limited to these:
  - Can you come up with a method to compare the performance of RR, FIFO, NORMAL and WRR?
  - Can you build WRR in a multi-cpu architecture and implement load balance?

# Hints

- To change the scheduler, study several functions with SYSCALL in their names. For example, SYSCALL\_DEFINE3 (sched\_setscheduler.....). Try to use these system calls in user space.
- You can firstly change scheduler to RR or FIFO to see if your testing file is logically correct.
- You can take full use of printk and the functions defined in /kernel/sched/debug.c for debugging.
- Helpful files:
  - /kernel/sched/core.c and /kernel/sched/sched.h tells you how the Linux scheduler works.
  - /kernel/sched/rt.c tells you how to create a scheduler.
  - /include/linux/sched.h concerns run-state processes.

Be patient enough to read them carefully!

# Hints

---

To install the *processtest.apk* to your AVD:

- Initiate your Android Virtual Device
- Type the command in Ubuntu Shell:
  - `adb install [the path of apk in Ubuntu (not in Android)]`
- Launch your apk by clicking the icon in Android AppList

# Report

---

- Explain how your `wrr.c` work.
- Explain what have you done in all the other files except `wrr.c`.
- Any further analysis is welcome.

# Something to Specify

- To give you an **overview** of this project:
  - You need to write 500 lines (more or less) of codes in:
    - `/kernel/sched/wrr.c`
  - You need to revise the following files to put *wrr.c* into effect:
    - `/arch/arm/configs/goldfish_armv7_defconfig`
    - `/include/linux/sched.h`
    - `/kernel/sched/sched.h`
    - `/kernel/sched/core.c`
    - `/kernel/sched/Makefile`
  - You need to write a test script to print the Scheduling Information of *processtest.apk* in foreground and background.

# Deadline

---

Mid-night, June 4, 2021

# Demo & Presentation

---

## ■ Demo:

- **June 5-6, 2021.** Demo slots will be posted in the WeChat group. Please sign your name in one of the available slots.

## ■ Presentation:

- You are encouraged to present your design of the project optionally. The presentation will be in the **afternoon of June 6, 2021.**



# 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/>