# OS 2017

Homework3: scheduling simulation

(Due date 12/28 23:59:59)

Lab @NCKU

# Objectives

- Simulate task scheduling
- Understand how to implement context switch
- Understand how signal works in Linux

# Requirements (1/2)

1. Write a user application (scheduling_simulator)
   - Shell mode
     - Implement 4 commands (*must follow the formats in slide 6*)
       - ***add***: Add new task(s)
       - ***remove***: Remove task(s)
       - ***ps***: Show the information of all tasks (PID, task name, task state, and queueing time)
       - ***start:*** Start or continue simulation (switch to simulation mode)
   - Simulation mode
     - Use ucontext and the related APIs to implement context switch
     - Implement the variable time quantum RR(round robin) scheduling
       - As in *slide 7*
       - Should receive a signal (SIGALRM) every 10 ms then determine whether to reschedule or not
     - ***Ctrl + z*** should pause the simulation and switch to shell mode
     - ***start*** should resume the simulation
       - continue simulation from where it pauses

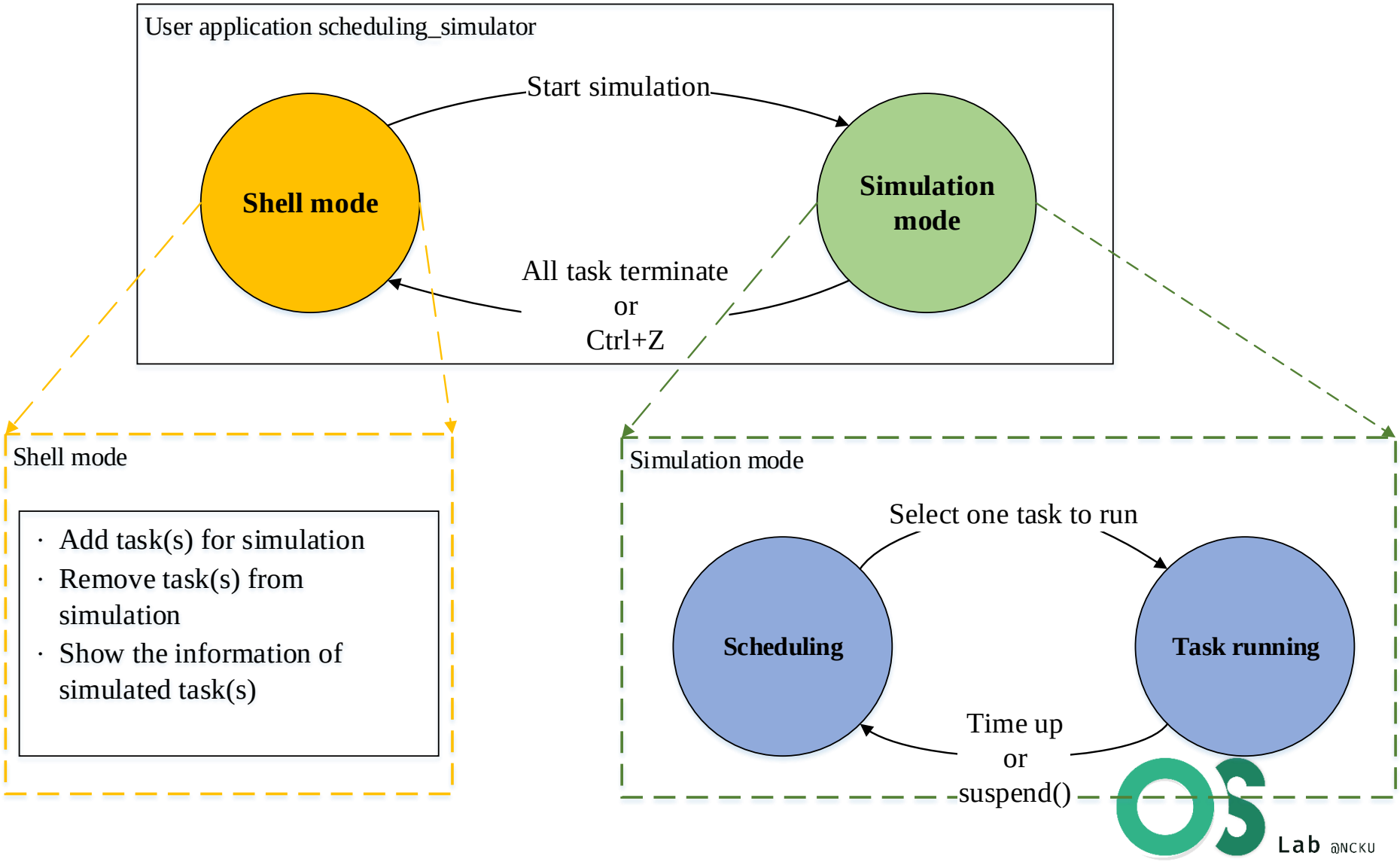
# Requirements (2/2)

2. Implement the APIs that can be used by the tasks (*described in slide 8*)
   - void hw_suspend(int msec_10);
   - void hw_wakeup_pid(int pid);
   - int hw_wakeup_taskname (char *task name);
   - int hw_task_create(char *task_name);

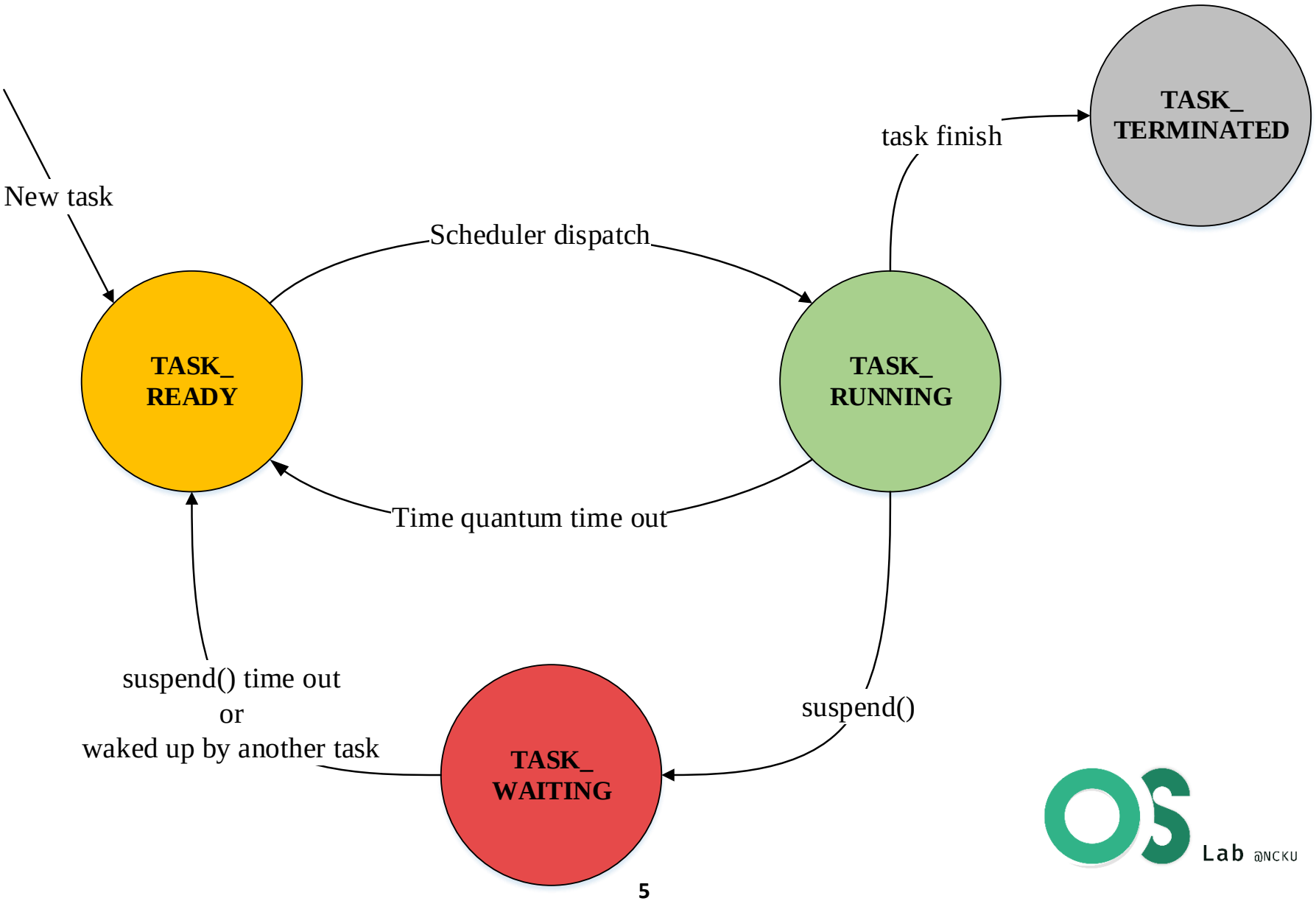
3. Task
   - The state of each task is shown in *slide 5*
   - A task is a function in ‘tasks.c’ (***task_name*** = function name)
   - All the functions are provided by TAs and can not be changed

Notice:
- Register signal handlers to handle ctrl+z and SIGALRM
- Signal may occur anytime even in signal handlers and APIs

# Architecture



# Task state


TASK_
TERMINATED
task finish
New task
Scheduler dispatch
TASK_
READY
TASK_
RUNNING
Time quantum time out
suspend() time out
or
waked up by another task
suspend()
TASK_
WAITING

# Shell commands

- add

$ add ***TASK_NAME*** -t ***TIME_QUANTUM***

- What task to add
- Optional argument
  - L for the larger time quantum
  - S for the small time quantum
- Default value is S

- remove

$ remove ***PID***

Remove a task with *PID*

- start

$ start

simulating…

- ps

simulating…

^Z

$ ps

| | | | |
|---|---|---|---|
| 1 | task1 | TASK_READY | 50 |
| 2 | task2 | TASK_TERMINATED | 10 |
| 3 | task2 | TASK_READY | 50 |
| 4 | task3 | TASK_WAITING | 50 |

PID — Task name — Task state (in slide p.5) — Queueing time

queueing time

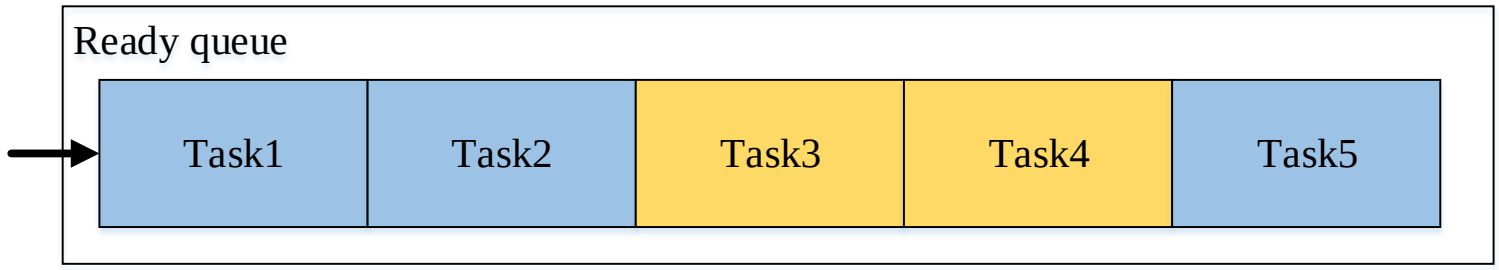
- The total time the task stays in the ready queue during all the simulation period

# Variable time quantum RR scheduling

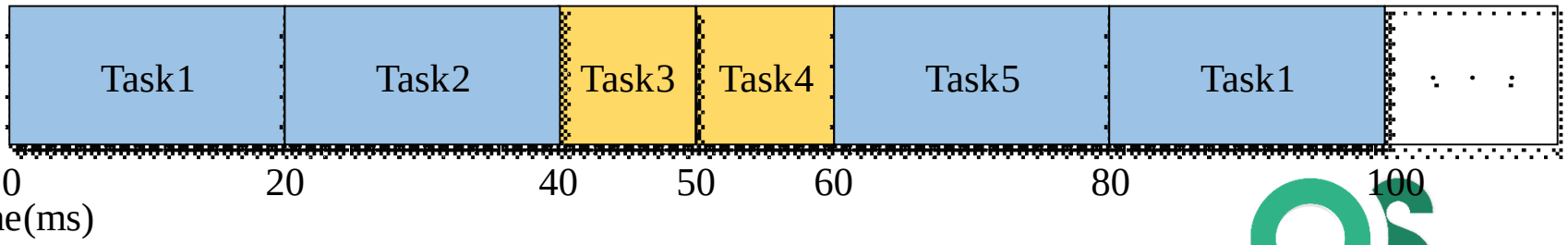
- Scheduling each task by round robin(RR)
- Two types of time quantum
  - Larger time quantum: 20 ms
  - Small time quantum: 10 ms

```
$ add Task1 -t L
$ add Task2 -t L
$ add Task3 -t S
$ add Task4
$ add Task5 -t L
$ start
```

Example




Gantt chart



# API Description

- void hw_suspend(int msec_10);
  - The running task change its state to ***TASK_WAITING***
  - Reschedule (schedule next task to run)
  - Change the state of the suspended task to ***TASK_READY*** after ***msec_10****10 ms
- void hw_wakeup_pid(int pid);
  - Change the state of task ***PID*** from ***TASK_WAITING*** to ***TASK_READY***
  - Reschedule if needed
- int hw_wakeup_taskname(char *task_name);
  - Change the state of all the tasks with ***task_name*** from ***TASK_WAITING*** to ***TASK_READY***
  - Return how many tasks are waken up
  - Reschedule if needed
- int hw_task_create(char *task_name);
  - Create task ***task_name***
  - Return ***PID*** of the created task
  - Return -1 if there is no function named ***task_name***
  - Reschedule if needed

# References


1. ucontext
   - The Open Group Library
   - IBM® IBM Knowledge Center
     - getcontext()
     - tetcontext()
     - makecontext()
     - swapcontext()

2. signal handler
   - Gitbook
   - Linux manual page

3. timer
   - Linux manual page
   - IBM® IBM Knowledge Center


Lab @NCKU