ECE 1175
Embedded Systems Design

Real-Time Scheduling - I

Wei Gao

The Scheduling Problem

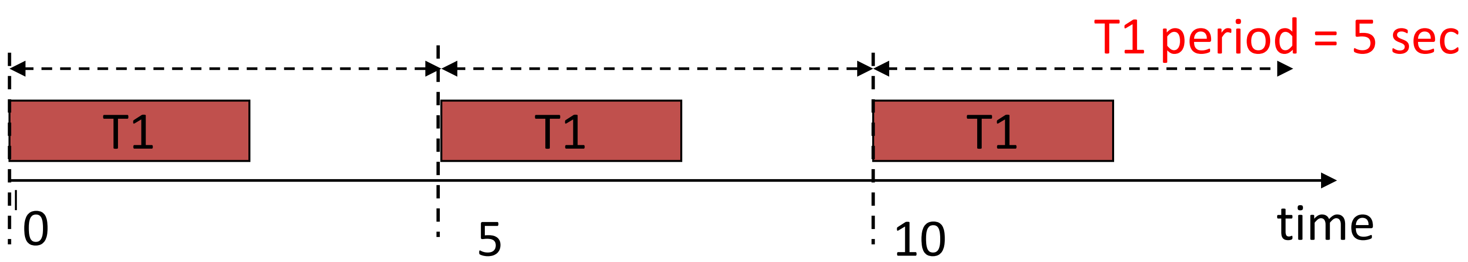
- Can we meet all deadlines?
  - Must be able to meet deadlines in all cases.
- How much CPU horsepower do we need to meet our deadlines?
- Timing violations: What happens if a process doesn’t finish by its deadline?
  - Hard deadline: system fails if missed.
  - Soft deadline: user may notice, but system doesn’t necessarily fail.

Process Scheduling: Embedded vs. General-Purpose

- General-purpose systems
  - e.g., PCs, database servers
  - Fairness to all tasks (no starvation)
  - Optimize throughput
  - Optimize average performance
- Embedded systems
  - Meet all deadlines.
  - Fairness or throughput is not important
  - Hard real-time: worry about worst case performance

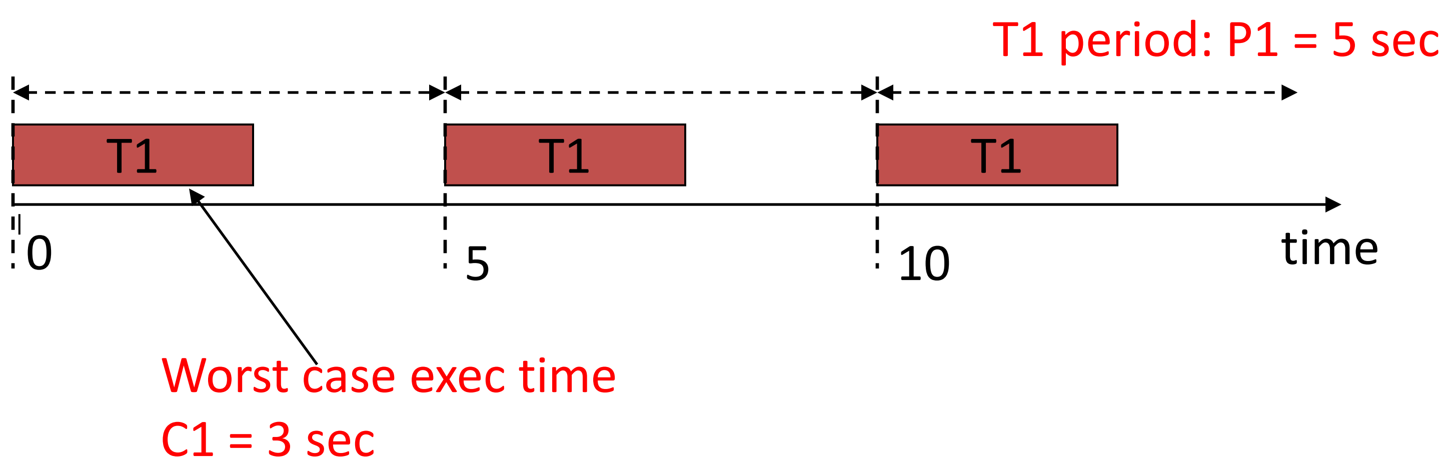
Terminologies Used in Scheduling

- Task
  - May correspond to a process or thread
  - May be released multiple times
- Periodic task
  - Ideal: inter-arrival time = period
  - General: inter-arrival time >= period
- Non-periodic task
  - Inter-arrival time does not have a lower bound
- Job: an instance of a task



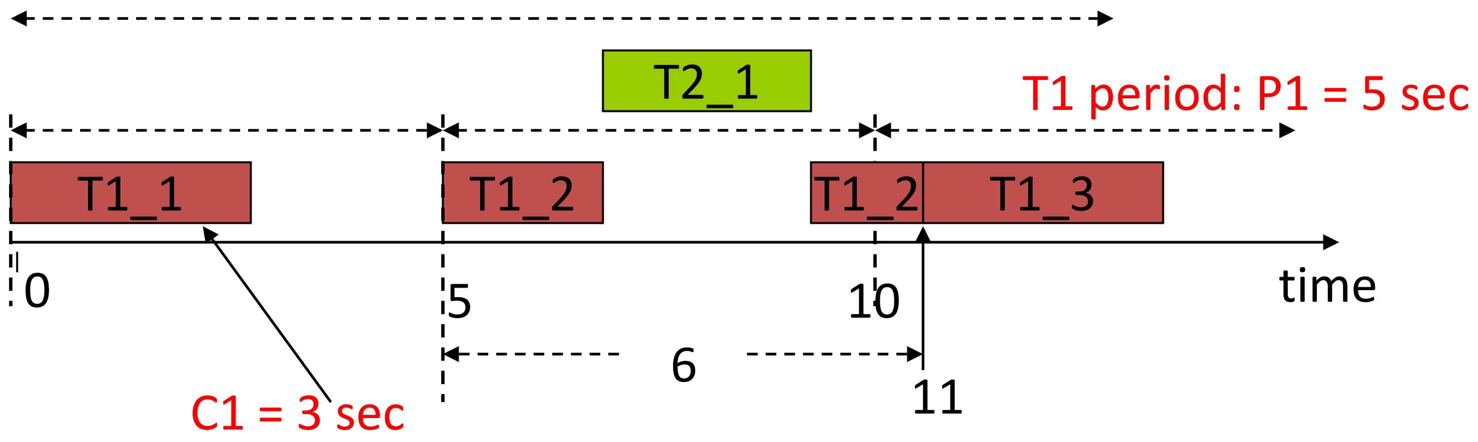
Timing Parameters - Task

- Task T_i
  - Period P_i
  - Worst-case execution time C_i
  - Relative deadline D_i
    - Usually equal to period



Timing Parameters - Job

- Job J_{ik} (denoted as Ti_k)
  - Release time: time when a job is ready
  - Response time R_i = finish time – release time
  - Absolute deadline = release time + D_i
- A job misses its deadline if
  - Response time $R_i > D_i$
  - Finish time > absolute deadline



Metrics to Evaluate Scheduling Algorithms

- Schedulability
  - A task set is schedulable under a scheduling algorithm if all jobs can meet their deadlines
- Overhead
  - Time required for scheduling decision and context switches.

Optimality

A scheduling algorithm S is optimal if

- a task set is not schedulable under S → it is not schedulable under any other algorithms

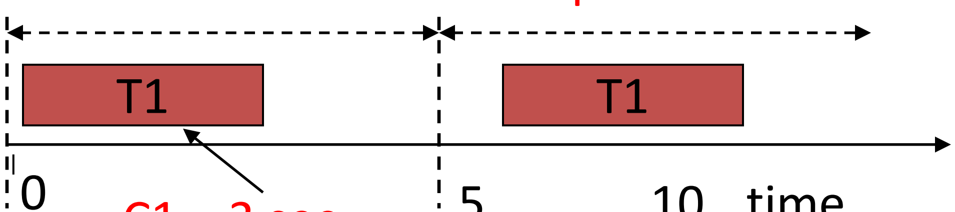
CPU Utilization Analysis

- Utilization of a processor:

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i}$$

n: number of tasks on the processor

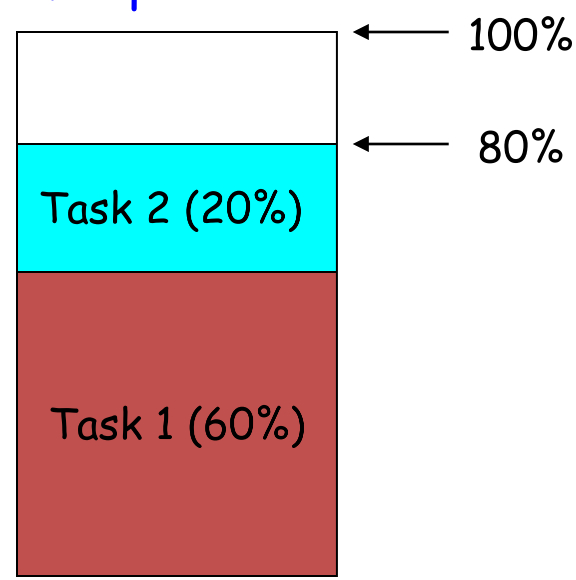
T1 period: P1 = 5 sec



C1 = 3 sec

CPU utilization of T1 = 3/5 = 60%

CPU utilization of a processor



Schedulable Utilization Bound

- Utilization bound U_b
  - All tasks are guaranteed to be schedulable if $U \leq U_b$
    - U is the requested utilization of a task set
- Conditions for scheduling
  - No scheduling algorithm can schedule a task set if $U > 1$
  - $U_b \leq 1$
  - An algorithm is optimal if its $U_b = 1$

Optimal Scheduling Algorithms

- Rate Monotonic Scheduling (RMS)
  - Higher rate (=1/period) → Higher priority
  - Optimal preemptive static priority scheduling algorithm
- Earliest Deadline First (EDF)
  - Earlier absolute deadline → Higher priority
  - Optimal preemptive dynamic priority scheduling algorithm

Assumptions

- Single processor.
- All tasks are periodic.
- Zero context switch time.
- Relative deadline = period.
- No priority inversion.

- RMS and EDF have been extended to cases with relaxed assumptions

RMS - Rate Monotonic Scheduling

- Common way to assign priorities
- Result from Liu & Layland, 1973 (JACM)
- Simple to understand and implement:

Processes with shorter period are given higher priority

- E.g.,

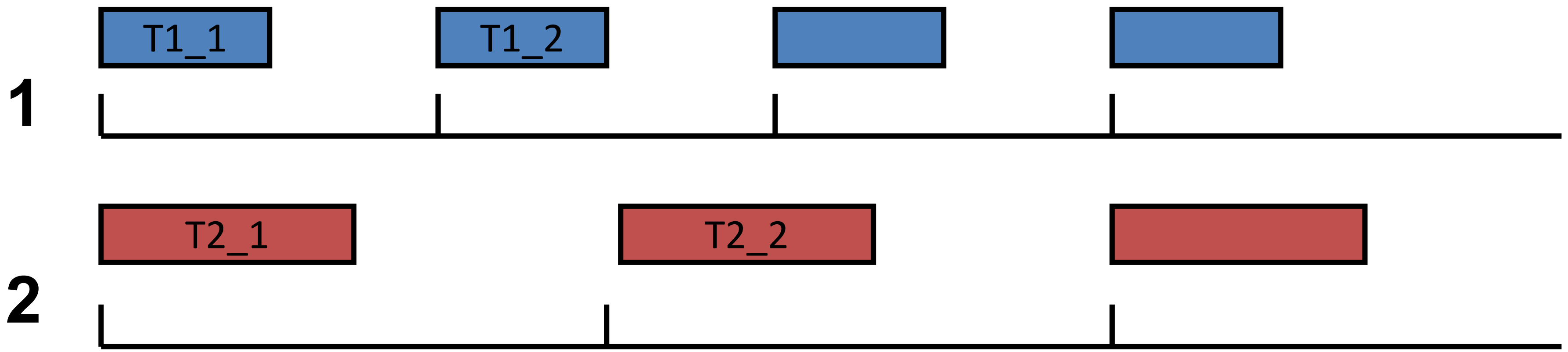
Period	Priority	
10	1	(highest)
12	2	
15	3	
20	4	(lowest)

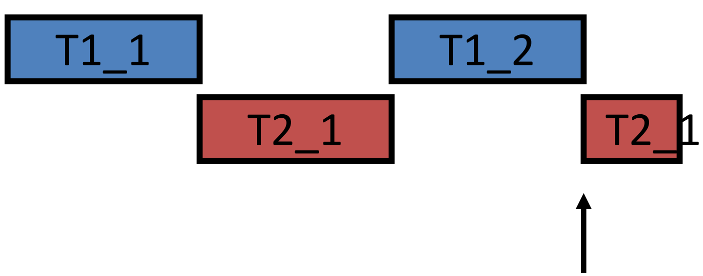
RMS Utilization Bound

- $U_b(n) = n(2^{1/n}-1)$
  - n: number of tasks
  - $U_b(1) = 1$
  - $U_b(2) = 0.828$
  - $U_b(n) \geq U_b(\infty) = \ln 2 = 0.693$
- $U \leq U_b(n)$ is a sufficient condition, but not necessary in general cases.
- U_b = 1 if all process periods are harmonic, i.e., periods are multiples of each other
  - e.g., 1,10,100

RMS Missing the Deadline

- T1 = (10,20), T2 = (15,30), utilization is 100% > RMS bound



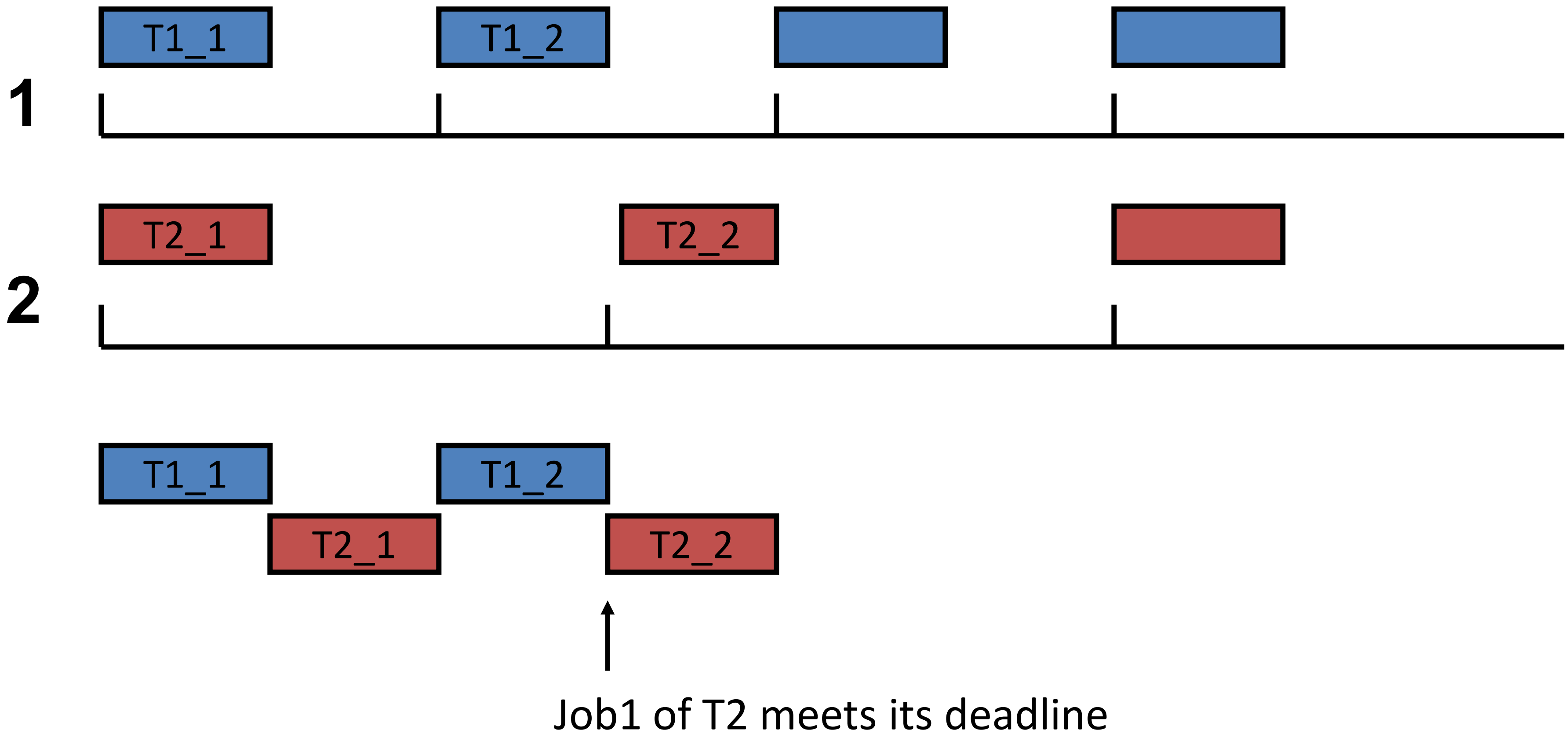


Would have met the deadline if T2 = (10,30), utilization reduced 83%

Job1 of T2 misses its deadline

RMS Meeting the Deadline

- T1 = (10,20), T2 = (**10**,30), utilization is **83%**

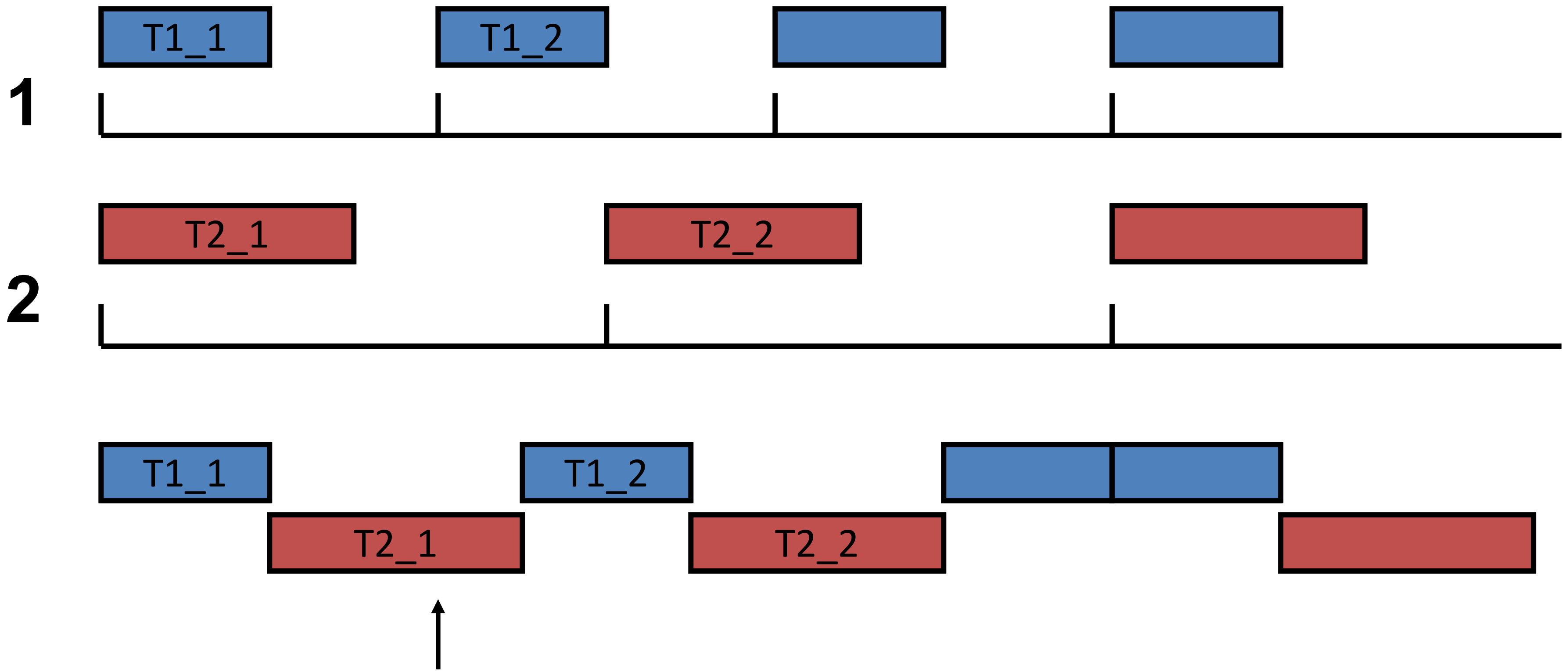


RMS Evaluation

- Schedulability
  - RMS may not guarantee schedulability even when CPU is not fully utilized
- Low overhead
  - When tasks are fixed, priorities are never changed
- Optimal static priority scheduling algorithm

EDF Meeting a Deadline

- T1 = (10,20), T2 = (15,30), utilization is 100%



T2 takes priority because its
deadline is sooner

EDF Utilization Bound

- $U_b = 1$
- $U \leq 1$ is a sufficient and necessary condition for schedulability.

- Generally considered too expensive to use in practice.
  - Absolute deadline for each job needs to be computed
  - Change process priorities on the fly

EDF Evaluation

- Schedulability
  - EDF can guarantee schedulability as long as CPU is not fully utilized
- Higher overhead than RMS
  - Task priorities may need to be changed online
- Optimal dynamic priority scheduling algorithm

Summary

- Terminologies and timing parameters
  - Task, job
- Metrics to evaluate scheduling algorithms
  - Schedulability, overhead
- Optimal scheduling algorithm: Rate Monotonic Scheduling (RMS)
  - Utilization bound
- Optimal scheduling algorithm: Earliest Deadline First (EDF)
  - Utilization bound, implementation, evaluation
- Relaxing assumptions: when relative deadline < period
  - Earliest Deadline First (EDF)
    - Processor demand analysis

Lab 4

- Lab 4 is due on 3/29
  - 6% in final grade
  - You will work on it on your own
    - No collaboration is allowed!
  - Need to let the TA check you off
- Real-Time Scheduling
  - http://www.pitt.edu/~weigao/ece1175/spring2021/lab4.htm