SUTD 51.505: 𝔽oundations of Cybersecurity (2018)

Exercise Sheet 4
October 5, 2018

- List your names (max 3 members for each group) on the answer sheet, **if you have actually worked on the exercises.**
- Answer questions in the same order as in the exercise sheet.
- Type in 12pt font, with 1.5 line spacing.
- There can be multiple acceptable answers. Justify carefully your reasoning.
- Go to the point, avoid copying verbatim definitions from the slides or the book.
- Submit your classwork and homework solutions (in pdf file) to eDimension by the deadlines below. Each group only needs one submission.
- Grading: total 100 points for each classwork and homework, each exercise has equal points in the same classwork and homework.

Classwork due on Friday October 5, 10:00 PM

Exercise 1
Identify and describe the possible time-of-check time-of-use (TOCTOU) errors in the following code snippet, and provide correction of the errors:

```
int addMoneyOnDate(String sourceAccount, String destinationAccount,
                   Money amount, Date date) {
  Date today = getCurrentDate();
  if (date.equals(today))
    if (currentUser.owns(sourceAccount))
      if (sourceAccount != destinationAccount) {
        File *sourceFile = new File(sourceAccount);
        File *destinationFile = new File(destinationAccount);
        if (balance(sourceFile) >= amount) {
          debitByAmount(sourceFile, amount);
          creditByAmount(destinationFile, amount);
          }
  }
}
```

Exercise 2
A computer has three commonly used resources designated A, B and C. Up to three processes designated X, Y and Z run on the computer and each makes periodic use of two of the three resources.

- Process X acquires A, then B, uses both and then releases both.
- Process Y acquires B, then C, uses both and then releases both.
- Process Z acquires C, then A, uses both and then releases both.

a) If two of these processes are running simultaneously on the machine, can a deadlock occur? If so, describe the deadlock scenario.

b) Describe a scenario in which deadlock occurs if all three processes are running simultaneously on the machine.

c) Modify the algorithm for acquiring resources so that deadlock cannot occur with three processes running.

Exercise 3

The following is a set of three interacting processes that can access two shared semaphores: U = 2; V = 1;

```
[Process 1]        [Process 2]         [Process 3]
 L1: wait(U)        L2: wait(V)         L3: wait(V)
     type("C")          type("A")           type("D")
     signal(V)          type("B")           goto L3
     goto L1            signal(V)
                        goto L2
```

Within each process the statements are executed sequentially, but statements from different processes can be interleaved in any order that's consistent with the constraints imposed by the semaphores. Assume that once execution begins, the processes will be allowed to run until all 3 processes are stuck in a wait() statement, at which point execution is halted.

a) Assuming execution is eventually halted, how many C's are printed when the set of processes runs? Why?

b) Assuming execution is eventually halted, how many D's are printed when this set of processes runs? Why?

c) Is CCDDABD a possible output sequence when this set of processes runs? Why?

Homework due on Friday October 12, 6:59 PM

Exercise 1

The *dining philosophers' problem* was proposed by Edgar Dijkstra and described here: `http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1000.html` The problem says that there are N of philosophers sitting at a round table and thinking about problems. Each philosopher has a plate of spaghetti in front of him/her and his/her own fork to the left of the plate. However, to eat spaghetti one has to use 2 forks at the same time, which means that a philosopher has to borrow the fork of his/her neighbor in order to eat. After eating, a philosopher puts both forks down in their corresponding places and then continues to think. As explained in the class, this will cause the deadlock problem. Provide a solution (with detailed steps) to address this problem.

Exercise 2

The*Byzantine generals' problem* was proposed by Lamport et al. in the following paper: `http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf` The problem says that a reliable computer system must be able to cope with the failure of one or more components. A failing component may exhibit a type of behavior that confuses other components, i.e. sending conflicting information to other components.

a) Describe an algorithm which can guarantee that all properly functioning components can reach a common decision/state given a certain number of failing components.

b) What is the maximum number of failing components that this algorithm can tolerate?

c) Why can't the algorithm tolerate more failing components?

Exercise 3

The following pair of processes share a common variable X, and use a shared binary semaphore S:

```
[Process A]                [Process B]
      int Y;                     int Z;
      wait(S);                   wait(S);
  A1: Y = X*2;               B1: Z = X+1;
  A2: X = Y;                 B2: X = Z;
      signal(S);                 signal(S);
```

S is set to 1 before either process begins execution and X is set to 5. Statements within a process are executed sequentially, but statements in process A may execute in any order with respect to statements in process B and vice versa. Each process will only execute once.

a) How many different values of X are possible after both processes finish executing? Why?

b) Suppose the programs are modified as follows to use a shared binary semaphore T:

```
[Process A]                [Process B]
      int Y;                     int Z;
                                 wait(T);
  A1: Y = X*2;               B1: Z = X+1;
  A2: X = Y;                 B2: X = Z;
      signal(T);
```

T is set to 0 before either process begins execution and, as before, X is set to 5. Now, how many different values of X are possible after both processes finish executing? Why?