VE482 Homework 4

Liu Yihao 515370910207

Ex. 1 — Simple questions

1.

2.

Ex. 2 — Monitors

The solution is very inefficient because when a process is blocked by the waituntil operation, the system must keeping evaluating the value of the boolean expression. This will cause much computing resource especially when the expression is complicated.

Ex. 3 — Race condition in Bash

1. ex3_race.sh

```
#!/bin/bash
if [ ! -f "ex3.txt" ]; then
    echo 0 > ex3.txt
fi
for ((i=0;i<100;++i)); do
    num=$(tail -1 ex3.txt)
    let "num=num+1"
    echo $num >> ex3.txt
done
```

   Use the command to execute it, the race condition will be found in the first output, we can observe two number "1".

```
bash ./ex3_race.sh & bash ./ex3_race.sh
```

2. ex3_no_race.sh

```
#!/bin/bash
if [ ! -f "ex3.txt" ]; then
    echo 0 > ex3.txt
fi
for ((i=0;i<100;++i)); do
    (
        flock 3
        num=$(tail -1 ex3.txt)
```

```
        let "num=num+1"
        echo $num >> ex3.txt
    )3<>ex3.txt
done
```

Use the command to execute it, no race condition is found.

```
bash ./ex3_no_race.sh & bash ./ex3_no_race.sh
```

Ex. 4 — Programming with semaphores

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define N 1000000
int count = 0;

void *thread_count(void *a) {
    int i, tmp;
    sem_t *sem = a;
    for (i = 0; i < N; i++) {
        sem_wait(sem);
        tmp = count;
        tmp = tmp + 1;
        count = tmp;
        sem_post(sem);
    }
}

int main(int argc, char *argv[]) {
    int i;
    sem_t sem;
    pthread_t *t = malloc(2 * sizeof(pthread_t));
    if (sem_init(&sem, 0, 1) != 0) {
        printf("ERROR init semaphore\n");
        exit(0);
    }
    for (i = 0; i < 2; i++) {
        if (pthread_create(t + i, NULL, thread_count, &sem)) {
            fprintf(stderr, "ERROR creating thread %d\n", i);
            exit(1);
        }
    }
    for (i = 0; i < 2; i++) {
        if (pthread_join(*(t + i), NULL)) {
            fprintf(stderr, "ERROR joining thread\n");
            exit(1);
        }
```

```
40      }
41      if (count < 2 * N) printf("Count is %d, but should be %d\n", count, 2 * N);
42      else printf("Count is[%d]\n", count);
43      free(t);
44      pthread_exit(NULL);
45  }
```