Σετ εργασιών 4

Ομάδα 6

Γιαννούκκος Παναγιώτης 2280

Χωροπανίτης Πασχάλης 2453

Άσκηση 4.1 - Σκέψη

- Αρχικά, δημιουργίσαμε ένα struct co_t όπου αποθηκεύει πληροφορία για το context και το stack του.
- Η διεπαφή mycoroutines.h υλοποιεί τις βασικές λειτουργίες init, create, switchto και destroy.
- Η init παίρνει το context της main για να το έχουμε ως σημείο επιστροφής.
- Η create αρχικοποιεί ένα context για την εκτέλεση μιας συνάρτησης.
- Η switchto καλεί τη swapcontext() για εναλλαγή.
- Η destroy απελευθερώνει τη μνήμη που δεσμεύτηκε για το struct co_t του κάθε context.

Άσκηση 4.1 - Υλοποίηση

```
co_t *main, *read, *write; char buffer[buffer_size];
```

write_context:

```
char c; int write_pos = 0;
while (read char from file) {
  buffer[write_pos] = c;
  write_pos++;
  if(buffer is full) {

    write_pos = 0;
    switchto(read);
  }
}

switchto(read);
```

read_context:

```
char c; int read_pos = 0;
while (1) {
  if(buffer[read_pos]==‘\0’)
     break;
  /* get char from buffer */
  /* write char to file */
  if(reached end of buffer){
     read_pos = 0;
     switchto(write);
}
```

Άσκηση 4.2 - Σκέψη

- Αρχικά φτιάξαμε ένα struct για το σηματοφόρο όπου κρατάει την τιμή του και ένα για το thread όπου κρατάει το context και το stack του.
- Επίσης χρησιμοποιούμε και μια κυκλική λίστα για να μπορεί ο scheduler να κάνει την εναλλαγή μεταξύ των threads.
- Ο scheduler καλείτε όταν κάποιος καλέσει yield() ή μέσω του handler.
- Ο handler καλείτε όταν πάρει SIGALRM signal.

Άσκηση 4.2 - Υλοποίηση

```
sem_t sem; thr_t **workers; data_t data[JOBS];
```

main:

```
/*get jobs from stdin*/
sem_init(sem, 1);
/*create workers*/
yield();
/*wait for workers to finish*/
/*destroy workers and sem*/
/*free allocated memory*/
```

worker:

```
while(1){
  sem_down(sem);
  if(all jobs done) break;
  /* get and process a job */
  sem_up(sem);
  yield();
}
```