Drugi kolokvijum iz Operativnih sistema 1
Maj 2024.

1. (10 poena)

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

#define handle_error(msg) \
  do { printf("Error: %s\n",msg); exit(-1); } while(0)

const int N = 10;

int main () {
  for (int i=0; i<N; i++) {
    pid_t pid = fork();
    if (pid<0) handle_error("Cannot create a process.");
    if (pid==0) {
      for (unsigned long j=0; j<(unsigned long)(i+1)*10000000; j++);
      exit(i);
    }
  }

  for (int i=0; i<N; i++) {
    int pi;
    pid_t pid = wait(&pi);
    printf("Child process #%d with pid=%d complete.\n",pi,pid);
  }

  return 0;
}
```

2. (10 poena)

```
Thread* Thread::parent = nullptr;
bool Thread::isActive = false;
Thread* Thread::awaitedChild = nullptr;
int Thread::status = 0;

void Thread::created () {
  this->parent = Thread::running;
  this->isActive = true;
  this->awaitedChild = nullptr;
  this->status = 0;
}

void Thread::completed (int status) {
  this->isActive = false;
  this->status = status;
  if (this->parent && this->parent->awaitedChild==this) {
    this->parent->awaitedChild = nullptr;
    Scheduler::put(this->parent);
  }
}

int Thread::join (int* status=nullptr) {
  lock();

  int ret = 0;
  Thread* thisThr = Thread::running;

  if (this->parent==thisThr)
    if (this->isActive)
      Thread::running->awaitedChild = child;
    else
      Scheduler::put(Thread::running);
  else {
    ret = -1;
    Scheduler::put(Thread::running);
  }

  Thread* newThr = Thread::running = Scheduler::get();
  yield(thisThr,newThr);
  if (ret==0 && status) *status = this->status;

  unlock();
  return ret;
}
```

3. (10 poena)

```
class GarageController {
public:
  GarageController (unsigned slots) { mySem = new Semaphore(slots); }
 ~GarageController () { delete mySem; }

  void entry () { mySem->wait(); }
  void exit () { mySem->signal(); }
  unsigned getFreeSlots () const { int n=mySem->val(); return (n>=0)?n:0; }

private:
  Semaphore* mySem;
};
```