Rešenja zadataka za
prvi kolokvijum iz Operativnih sistema 1
Odsek za računarsku tehniku i informatiku
April 2016.

1. (10 poena)

```
IORequest* ioPending = 0; // Currently pending request

void performIO () {
  if (ioHead!=0 && ioPending==0) {
    ioPending = ioHead; // Take the first request,
    ioHead = ioHead->next; // remove it from the list,
    if (ioHead==0) ioTail = 0;
    *ioCtrl = START_SENDING; // and send it to I/O
    for (int i=0; i<BLOCK_SIZE; i++)
      *ioData = ioPending->buffer[i];
    *ioCtrl = END_SENDING;
  }
}

void transfer (IORequest* req) {
  req->next = 0;
  if (!ioHead) {
    ioHead = ioTail = req;
    performIO();
  } else
    ioTail = ioTail->next = req;
}

interrupt void ioInterrupt () {
  if (ioPending==0) return; // Exception
  if (*ioStatus&1) // Error in I/O
    ioPending->status = -1;
  else
    ioPending->status = 0;
  ioPending = 0;
  performIO();
}
```

2. (10 poena)

U klasu `Thread` dodati su sledeći privatni, nestatički podaci-članovi sa datim inicijalnim vrednostima:

```
Thread* Thread::parent = 0;
bool Thread::isWaitingForChildren = false;
unsigned long Thread::activeChildrenCounter = 0;

void Thread::created (Thread* parent) {
  this->parent = parent;
  if (parent) parent->activeChildrenCounter++;
}

void Thread::completed () {
  if (this->myParent && --this->myParent->activeChildrenCounter==0 &&
      this->myParent->isWaitingForChildren) {
    this->myParent->isWaitingForChildren = false;
    Scheduler::put(this->myParent);
  }
}

void Thread::wait () {
  lock();
  jmp_buf old = Thread::running->context;
  if (Thread::running->activeChildrenCounter>0)
    Thread::running->isWaitingForChildren = true;
  else
    Scheduler::put(Thread::running);
  Thread::running = Scheduler::get();
  jmp_buf new = Thread::running->context;
  yield(old,new);
  unlock();
}
```

3. (10 poena)

```
int multiexec (int number, const char* filename, const char* const args[]){
  int ret = 0;
  // Prepare the arguments for the children:
  const char* childArgs[3];
  childArgs[0] = filename;
  childArgs[2] = NULL;
  for (int i=0; i<number; i++) {
    childArgs[1] = args[i];
    // Create a child:
    int status = fork();
    if (status<0) continue; // fork failed
    if (status==0) // Child's context
      if (execvp(filename,childArgs)<0) exit();
    else  // Parent's context
      ret++;
  }
  return ret;
}
```