Rešenja zadataka za
drugi kolokvijum iz Operativnih sistema 1
Jun 2017.

1. (10 poena) U klasu `Thread` treba dodati sledeći član:

```
Barrier* blockedOnBarrier(0);
```

```
class Barrier {
public:
  Barrier (int open=1) : isOpen(open), owner(Thread::running) {}
  void open();
  void close();
  void pass();
private:
  int isOpen;
  Thread* owner;
};

void Barrier::pass () {
  if (this->owner!=Thread::running) return;
  lock();
  if (!this->isOpen) {
    Thread::running->blockedOnBarrier = this;
    if (setjmp(Thread::running->context)==0) {
       Thread::running = Scheduler->get();
       longjmp(Thread::running->context,1);
    }
  }
  unlock();
}

void Barrier::open () {
  lock();
  this->isOpen = 1;
  if (this->owner->blockedOnBarrier==this) {
    this->owner->blockedOnBarrier = 0;
    Scheduler::put(this->owner);
  }
  unlock();
}

inline void Barrier::close () {
  this->isOpen = 0;
}
```

2. (10 poena)

```
union OverlaidData {
  struct Phase1Data p1data;
  struct Phase2Data p2data;
} data;
struct CommonData cdata;

int main () {
  FILE fd1 = fopen("overlay1.tmp");
  if (fd1<0) return fd1;
  FILE fd2 = fopen("overlay2.tmp");
  if (fd2<0) {
    fclose(fd1);
    return fd2;
  }

  int ret = 0;

  init_cdata(&cdata);
  init_p1data(&data.p1data);
  process_phase_1(&data.p1data,&cdata);
  if ((ret=fwrite(fd1,0,sizeof(data.p1data),&data.p1data))<0) {
    fclose(fd1);
    fclose(fd2);
    return ret;
  }

  init_p2data(&data.p2data);
  process_phase_2(&data.p2data,&cdata);

  while (!cdata.completed) {
    if ((ret=fwrite(fd2,0,sizeof(data.p2data),&data.p2data))<0) break;

    if ((ret=fread(fd1,0,sizeof(data.p1data),&data.p1data))<0) break;
    process_phase_1(&data.p1data,&cdata);
    if ((ret=fwrite(fd1,0,sizeof(data.p1data),&data.p1data))<0) break;

    if ((ret=fread(fd2,0,sizeof(data.p2data),&data.p2data))<0) break;
    process_phase_2(&data.p2data,&cdata);
  }

  fclose(fd1);
  fclose(fd2);
  return ret;
}
```

3. (10 poena) a)(3) VA(32): Page1(10):Page2(10):Offset(12).
PA(28): Frame(16):Offset(12).

b)(7)

```
const unsigned offsw = 12;

int resolvePageFault (PCB* pcb, unsigned long addr, SegDesc** ret) {
  if (pcb==0) return ERR_FATAL; // Exception: fatal exception!
  unsigned long page = addr>>offsw;
  SegDesc* sd = pcb->segDesc;
  while (sd!=0) {
    if (page>=sd->startingPage && page<sd->startingPage+sd->size) {
      *ret = sd;
      return LOAD_PAGE;
    }
    sd = sd->next;
  }
  return MEM_ACCESS_FAULT;
}
```