Rešenja trećeg kolokvijuma iz Operativnih sistema 2 Januar 2012.

1. (10 poena)

```
int vm_alloc (int pg, int sz) {
  static vm_area_desc vm;
  static vm_area_desc* ptr=&vm;
  vm.page=pg;
  vm.size=sz;
  asm {
    load r1,#0x21
    load r2,ptr
    int  0x31
  }
}
```

2. (10 poena)

```
#!/bin/bash
if test ! $# -eq 2
then
  echo "Nije prosledjen odgovarajuci broj parametara"
  exit 1
fi
if test ! -f $2 -o ! -w $2
then
  echo "Fajl $2 ne postoji ili nemate dozvolu za upis"
  exit 2
fi
if test ! -d $1
then
  echo "Direktorijum $1 ne postoji"
  exit 3
fi
ls $1 | sed "s/\(.\).*_\(.\).*_\(.*\)\.zip/\2\1\3d@student.etf.bg.ac.rs/"
>$2
```

3. (10 poena)

```
void acquireForksForPhilosopher(int *forks[N], int id, int msgQueueId) {
 forks[id] = 0;
 forks[(id + 1) % N] = 0;
 struct requestMsg msg_buf;
 msg_buf.mtype = id + 1;
 msg_buf.msg[0] = 1;
 msgsnd(msgQueueId, &msg_buf, sizeof(char), 0);
}

int main() {
 int requestMsgQueueId = msgget(MESSAGE_Q_KEY, IPC_CREAT | 0666);
 int responseMsgQueueId = msgget(MESSAGE_Q_KEY + 1, IPC_CREAT | 0666);
 size_t len = sizeof(char);

 //philosophers
```

```
 int id;
 for (id = 1; id <= N; id++) { // rezervisana vrednost za mtype=0
  if (fork() == 0) {
   philosopher(id);
  }
 }

 //waiter
 int *forks[N], *requests[N];
 for (id = 0; id < N; id++) {
  forks[id] = 1;
  requests[id] = 0;
 }

 struct requestMsg msg_buf;
 while (1) {
  int r = msgrcv(requestMsgQueueId, &msg_buf, len, 0, 0);
  id = (int) msg_buf.mtype - 1;

  if (msg_buf.msg[0] == 1) { //request forks
   if (forks[id] && forks[(id + 1) % N]) {
    acquireForksForPhilosopher(forks, id, responseMsgQueueId);
   } else {
    requests[id] = 1;
   }
  } else { //Release forks
   forks[id] = 1;
   forks[(id + 1) % N] = 1;

   // check neighbors
   int leftNeighbour = id ? id - 1 : N - 1;
   int rightNeighbour = (id + 1) % N;
   if (requests[rightNeighbour] && forks[(rightNeighbour + 1) % N]) {
    requests[rightNeighbour] = 0;
    acquireForksForPhilosopher(forks, rightNeighbour,
      responseMsgQueueId);
   }
   if (requests[leftNeighbour] && forks[leftNeighbour]) {
    requests[leftNeighbour] = 0;
    acquireForksForPhilosopher(forks, leftNeighbour,
      responseMsgQueueId);
   }
  }
 }
}
```