Rešenja trećeg kolokvijuma iz Operativnih sistema 2, Januar 2014.

1. (10 poena)

```
extern char* CLI_path;
int execute_command(char* command_name, char** args){
  static create_process_struct ps;
  static create_process_struct* ptr=&ps;
  ps.program_file = strcat(CLI_path, command_name);
  ps.args = args;
  asm {
    load r1,#0x31
    load r2,ptr
    int  0x11
  }
}
```

2. (10 poena)

```
#!/bin/bash

if [ $# -lt 3 ];then
    echo 'Error: Insufficient arguments.'
    exit 1
fi

TMP='tmp123'
FILE=$1
shift
USER=$1
shift
grep "^$USER" $FILE > /dev/null
if [ $? -eq 0 ];then
      while [ $# -gt 0 ];do
            grep "^$USER.*$1" $FILE > /dev/null
            if [ $? -ne 0 ];then
                  cat $FILE | sed "s:\(^$USER.*\):\1 $1:" > $TMP
                  cat $TMP > $FILE
                  rm $TMP
            fi
            shift
      done
else
      echo "$USER $@" >> $FILE
fi
```

3. (10 poena)

```
void giveTokenToClient(int id, int responseMsgQueueId) {
 struct requestMsg msg_buf;
 msg_buf.mtype = id + 1;
 msg_buf.msg[0] = 1;
 msgsnd(responseMsgQueueId, &msg_buf, sizeof(char), 0);
}

int main(int argc, const char **argv) {
 int M,N;
 if ( argc > 2 ) {
      M = atoi( argv[1] );
      N = atoi( argv[2] );
    }
 else return -1;
 int requestMsgQueueId = msgget(MESSAGE_Q_KEY, IPC_CREAT | 0666);
 int responseMsgQueueId = msgget(MESSAGE_Q_KEY + 1, IPC_CREAT | 0666);
 size_t len = sizeof(char);

 //clients
 int id;
 for (id = 1; id <= N; id++) {
  if (fork() == 0) {
   client(id);
  }
 }
 //server
 int requests[N];
 int tokens = M, waiting = 0;
 for (id = 0; id < N; id++) {
  requests[id] = 0;
 }

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

  if (msg_buf.msg[0] == 1) { //request token
   if (tokens > 0) {
    tokens--;
    giveTokenToClient(id, responseMsgQueueId);
   } else {
    requests[id] = 1;
    waiting++;
   }
  } else { //release token
   if(waiting > 0)
   {
    int max = 0, maxId = 0; //avoid starvation - aging
    for (id = 0; id < N; ++id) {
     if (requests[id] > max) {
      max = requests[id];
      maxId = id;
     }
     if(requests[id])
      requests[id]++;
    }
    requests[maxId] = 0;
    giveTokenToClient(maxId, responseMsgQueueId);
    waiting --;
   }
   else tokens++;
  }
 }
}
```