Rešenja zadataka sa trećeg kolokvijuma iz Operativnih sistema 2, januar 2018.

1. (10 poena)

```
int recoverBlock (unsigned dsk, unsigned long blk, byte* buffer) {
  byte auxBuf[BlockSize]; // Auxiliary buffer
  for (int i=0; i<BlockSize; i++)
    buffer[i] = 0;
  for (unsigned d=0; d<=StripeSize; d++) {
    if (d==dsk) continue; // Disk dsk is out of order, skip it
    int status = readBlock(d,blk,auxBuf);
    if (status!=0) return status;
    for (int i=0; i<BlockSize; i++)
      buffer[i] ^= auxBuf[i];
  }
  return 0;
}
```

2. (10 poena)

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Wrong number of arguments"
    exit 1
fi

command=$1
dir=$2

pid=$(ps a | grep $command | grep -v grep | tr -s " " | cut -d" " -f2)

if kill $pid; then
    pushd $dir
    if make; then
      popd
      $command
    else
      echo "Compilation fails"
      exit 1
    fi
else
    echo "Kill command failed"
fi
```

3. (10 poena)

```
const char DEVICE_NAME[] = "/dev/ud1";
const size_t DEVICE_SIZE = 1 << 30;
const int MSG_KEY = 1;
void *open_device(int *file) {
    int fd = open(DEVICE_NAME, O_WRONLY);
    void *mem_pointer =
          mmap(0, DEVICE_SIZE, PROT_WRITE, MAP_SHARED, fd,0);
    *file = fd;
    return mem_pointer;
}
void close_device(int file, void *address) {
    munmap(address, DEVICE_SIZE);
    close(file);
}
void write_io(char *device_address, int address,
              char *data, int size) {
    for (int i = 0; i < size; i++) {
        device_address[address + i] = data[i];
    }
}
struct msg_t {
    int size;
    int address;
    char data[128];
};
struct msg_buf {
    long mtype;
    struct msg_t data;
};
int main() {
    int msg_box = msgget(MSG_KEY, IPC_CREAT | 0666);
    int fd;
    void *device_address;
    device_address = open_device(&fd);
    while (1) {
        struct msg_buf msg;
        msgrcv(msg_box, &msg, sizeof(struct msg_t), 1, 0);
        if (msg.data.size == 0) {
            break;
        }
        write_io(device_address, msg.data.address,
                 msg.data.data, msg.data.size);
    }
    close_device(fd, device_address);
    msgctl(msg_box, IPC_RMID, 0);
    return 0;
}
```