Rešenja zadataka za
treći kolokvijum iz Operativnih sistema 1
Jun 2022.

1. (10 poena)

```
class KeyboardBuffer {
public:
  KeyboardBuffer() : head(0),tail(0),count(0),mutex(1),itemAvailable(0) {}
  char getc ();
  void putc (char c);

private:
  static const size_t KB_SIZE = 256;
  char buffer[KB_SIZE];
  size_t head, tail, count;
  Semaphore mutex, itemAvailable;
};

static KeyboardBuffer* instance () {
  static KeyboardBuffer _instance;
  return &_instance;
}

char KeyboardBuffer::getc () {
  itemAvailable.wait();
  mutex.wait();
  kbint_mask();
  char c = buffer[head];
  head = (head+1)%KB_SIZE;
  count--;
  kbint_unmask();
  mutex.signal();
  return c;
}

void KeyboardBuffer::putc (char c) {
  if (count<KB_SIZE) {
    buffer[tail] = c;
    tail = (tail+1)%KB_SIZE;
    count++;
    itemAvailable.signal();
  }
}

interrupt void keyb_int () {
  while (*ioStatus & 1)
    KeyboardBuffer::instance()->putc(*ioData);
}
```

2. (10 poena)

```
#include <fcntl.h>

int readSubtree (int fd, Node** node) {
  static NodeData data;
  static int left, right;
```

```
  ssize_t cnt = read(fd,&data,sizeof(NodeData));
  if (cnt<sizeof(NodeData)) return -2;
  cnt = read(fd,&left,sizeof(int));
  if (cnt<sizeof(int)) return -2;
  cnt = read(fd,&right,sizeof(int));
  if (cnt<sizeof(int)) return -2;

  *node = new Node();
  if (*node == 0) return -3;
  (*node)->data = data;
  (*node)->left = (*node)->right = 0;

  int r = 0;
  if (left) {
    r = readSubtree(fd,&((*node)->left));
    if (r<0) return r;
  }
  if (right) {
    r = readSubtree(fd,&((*node)->right));
    if (r<0) return r;
  }
  return 0;
}

int readTree (const char* pathname, Node** root) {
 int fd = open(*pathname,O_RDONLY);
 if (fd<0) return -1;

 int r = readSubtree(fd,root);

 close(fd);
 return r;
}
```

3. (10 poena)

```
class FLogicalAccess {
public:

  FLogicalAccess () { reset(0,0); }
  void reset (size_t offset, size_t size);
  int  end() const { return isEnd; }
  void next();

  size_t getBlock() const { return blk; }
  size_t getRelOffset() const { return relOfs; }
  size_t getRelSize() const { return relSz; }

private:
  size_t size;
  size_t blk, relOfs, relSz;
  int isEnd;
};

inline void FLogicalAccess::reset (size_t offset, size_t sz) {
    blk = offset/BLK_SIZE;
    relOfs = offset%BLK_SIZE;
    relSz = (sz>BLK_SIZE-relOfs)?(BLK_SIZE-relOfs):sz;
    isEnd = (sz==0);
    size -= relSz;
}
```

```
inline void FLogicalAccess::next () {
  if (size>0) {
    blk++;
    relOfs = 0;
    relSz = (size>BLK_SIZE)?BLK_SIZE:size;
    isEnd = 0;
    size -= relSz;
  }
  else reset(0,0);
```