


## TEMA 2

### CERINTA 2

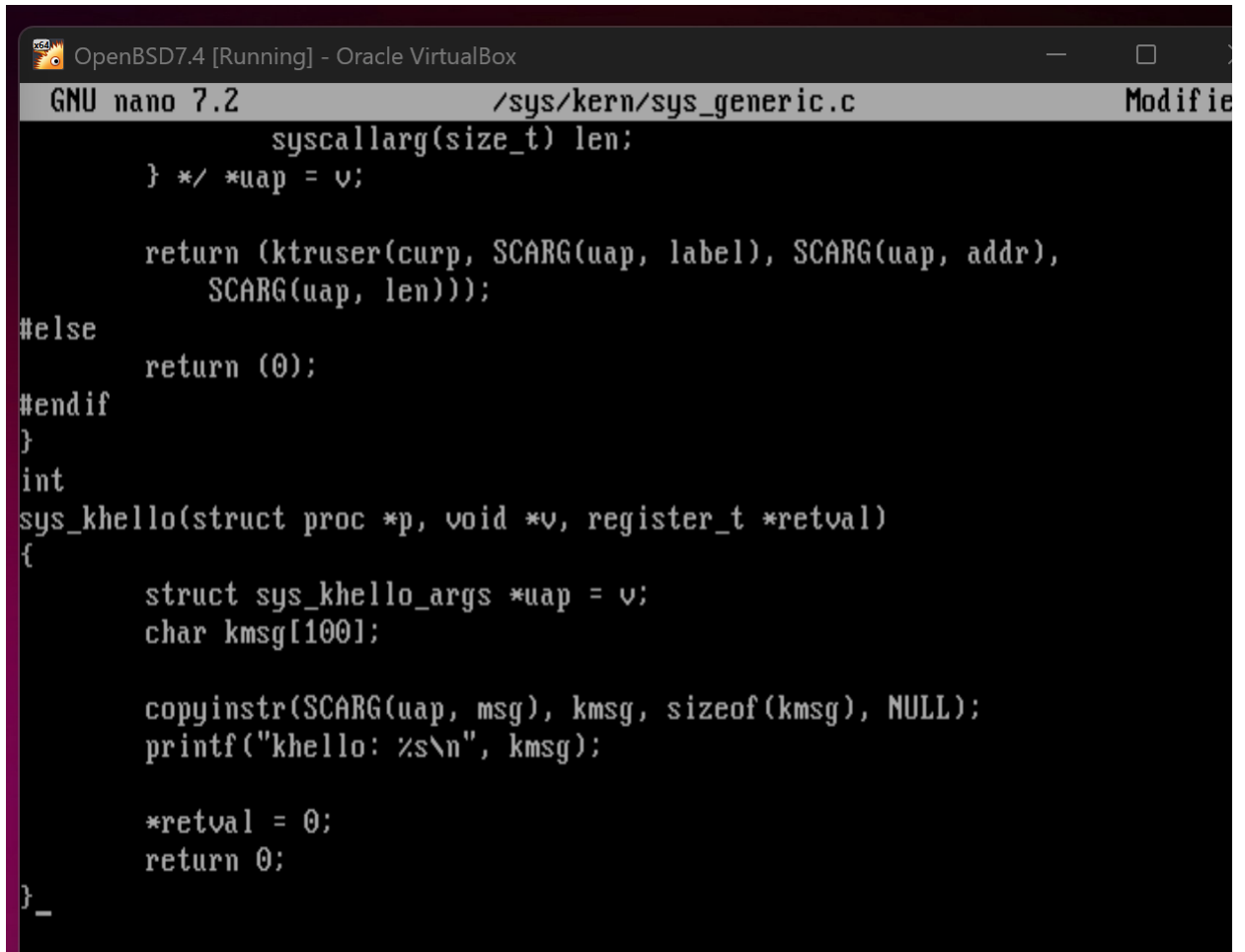


```
GNU nano 7.2 /sys/kern/syscalls.master Modified
mode_t mode); }
319 STD { int sys_mkfifoat(int fd, const char *path, \
mode_t mode); }
320 STD { int sys_mknodat(int fd, const char *path, \
mode_t mode, dev_t dev); }
321 STD { int sys_openat(int fd, const char *path, int flags, \
... mode_t mode); }
322 STD { ssize_t sys_readlinkat(int fd, const char *path, \
char *buf, size_t count); }
323 STD { int sys_renameat(int fromfd, const char *from, \
int tofd, const char *to); }
324 STD { int sys_symlinkat(const char *path, int fd, \
const char *link); }
325 STD { int sys_unlinkat(int fd, const char *path, \
int flag); }
326 OBSOL t32_utimensat
327 OBSOL t32_futimens
328 OBSOL __tfork51
329 STD NOLOCK { void sys___set_tcb(void *tcb); }
330 STD NOLOCK { void *sys___get_tcb(void); }
331 STD { int sys_khello(const char*msg);}

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^_ Replace ^U Paste ^J Justify ^_ Go To Line
```

- ⇒ Adaug semnatura primei functii la primul id liber din /sys/kern/syscalls.master, 331
- ⇒ Iau ca argument un ptr la un string





The screenshot shows a terminal window titled "OpenBSD7.4 [Running] - Oracle VirtualBox". Inside, the GNU nano 7.2 editor is open, editing the file /sys/kern/sys\_generic.c. The code is as follows:

```
        syscallarg(size_t) len;
    } /* *uap = v;

    return (ktruser(curp, SCARG(uap, label), SCARG(uap, addr),
        SCARG(uap, len)));
#else
    return (0);
#endif
}

int
sys_khello(struct proc *p, void *v, register_t *retval)
{
    struct sys_khello_args *uap = v;
    char kmsg[100];

    copyinstr(SCARG(uap, msg), kmsg, sizeof(kmsg), NULL);
    printf("khello: %s\n", kmsg);

    *retval = 0;
    return 0;
}
```

- ⇒ Adaug codul in c in sys/kern/sys\_generic.c
- ⇒ Ma folosesc de SCARG ca sa extrag ptr ul de la msg
- ⇒ Il afisez cu un printf
- ⇒ Recompiliez kernel ul si dau reboot
- ⇒



```
labSO# cat test_funcție.c
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>

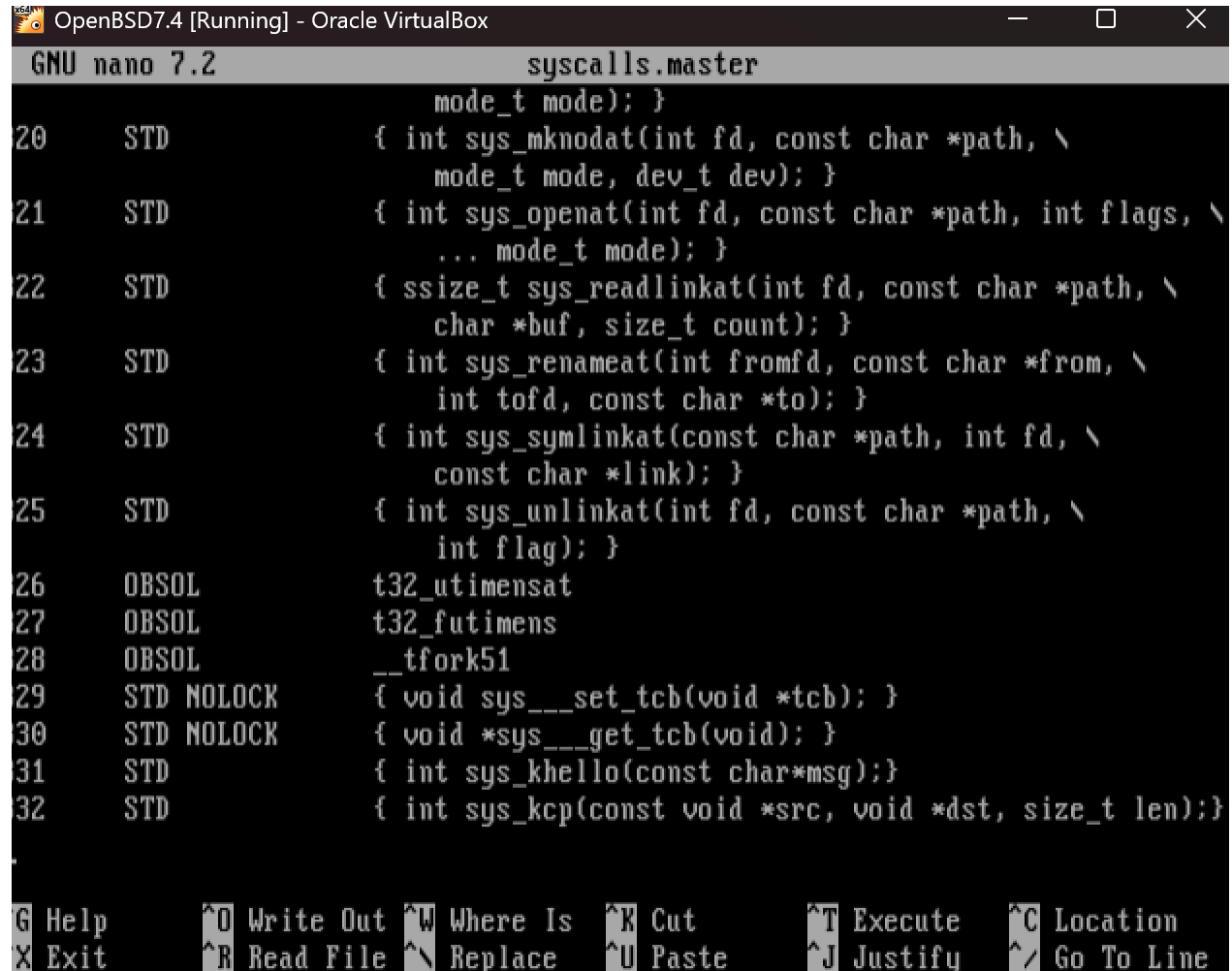
int main() {
    syscall(331, "world");
    return 0;
}
labSO# ./test_funcție.c
ksh: ./test_funcție.c: cannot execute - Permission denied
labSO# cat test_funcție.c
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>

int main() {
    syscall(331, "world");
    return 0;
}
labSO# ./test_funcție
khello: world
labSO#
```

- ⇒ Testez codul cu un c file in care fac syscall 331 si ii dau string ul world si el imi va afisa “khello: world” folosindu se de functia din kernel



## CERINTA 3

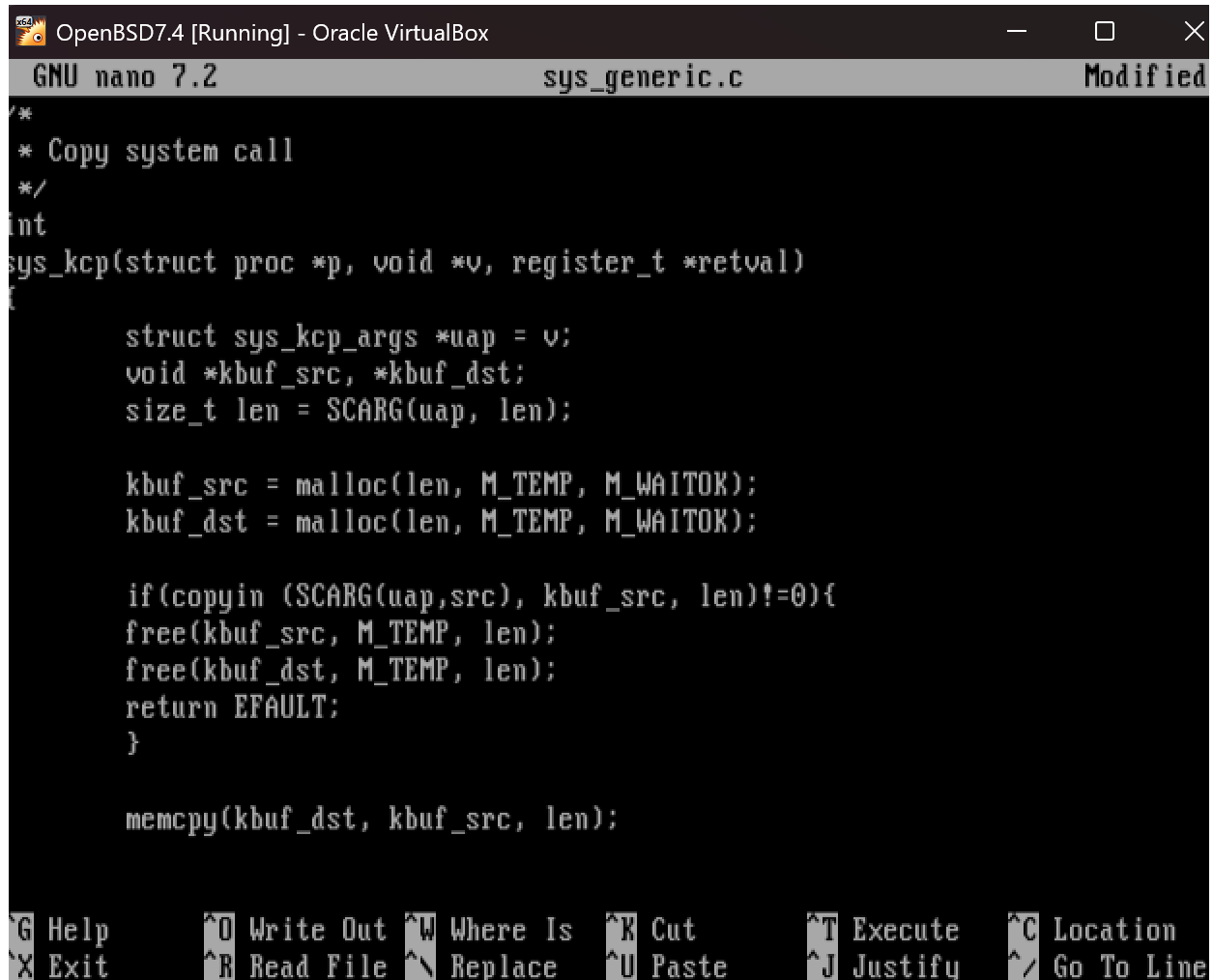


The screenshot shows a window titled "OpenBSD7.4 [Running] - Oracle VirtualBox" containing a nano 7.2 editor. The editor is editing a file named "syscalls.master". The content of the file is a list of system calls with their categories and prototypes. The categories are listed in the first column, followed by a tab, then the category name (STD, OBSOL, or NOLOCK), another tab, and finally the function prototype. The functions are: sys\_mknodat, sys\_openat, sys\_readlinkat, sys\_renameat, sys\_symlinkat, sys\_unlinkat, t32\_utimensat, t32\_futimens, \_\_tfork51, sys\_\_\_set\_tcb, sys\_\_\_get\_tcb, sys\_khello, and sys\_kcp. The bottom of the window shows the nano editor's command palette with various shortcuts like ^G for Help, ^O for Write Out, etc.

```
GNU nano 7.2 syscalls.master
mode_t mode); }
20  STD      { int sys_mknodat(int fd, const char *path, \
mode_t mode, dev_t dev); }
21  STD      { int sys_openat(int fd, const char *path, int flags, \
... mode_t mode); }
22  STD      { ssize_t sys_readlinkat(int fd, const char *path, \
char *buf, size_t count); }
23  STD      { int sys_renameat(int fromfd, const char *from, \
int tofd, const char *to); }
24  STD      { int sys_symlinkat(const char *path, int fd, \
const char *link); }
25  STD      { int sys_unlinkat(int fd, const char *path, \
int flag); }
26  OBSOL    t32_utimensat
27  OBSOL    t32_futimens
28  OBSOL    __tfork51
29  STD NOLOCK { void sys___set_tcb(void *tcb); }
30  STD NOLOCK { void *sys___get_tcb(void); }
31  STD      { int sys_khello(const char*msg);}
32  STD      { int sys_kcp(const void *src, void *dst, size_t len);}
.
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^\_ Replace ^U Paste ^J Justify ^\_ Go To Line





```
OpenBSD7.4 [Running] - Oracle VirtualBox
GNU nano 7.2 sys_generic.c Modified
/*
 * Copy system call
 */
int
sys_kcp(struct proc *p, void *v, register_t *retval)
{
    struct sys_kcp_args *uap = v;
    void *kbuf_src, *kbuf_dst;
    size_t len = SCARG(uap, len);

    kbuf_src = malloc(len, M_TEMP, M_WAITOK);
    kbuf_dst = malloc(len, M_TEMP, M_WAITOK);

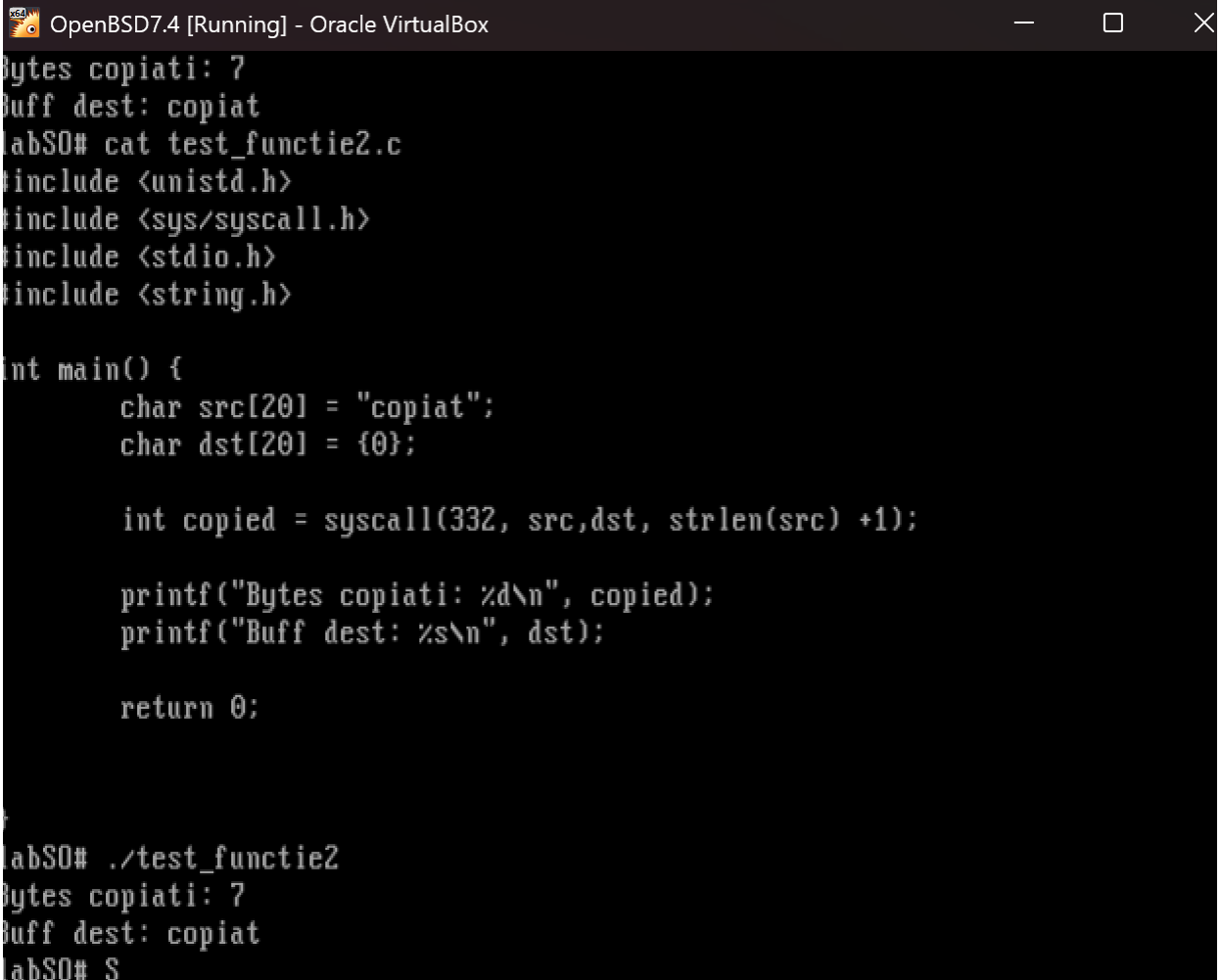
    if(copyin(SCARG(uap, src), kbuf_src, len) != 0){
        free(kbuf_src, M_TEMP, len);
        free(kbuf_dst, M_TEMP, len);
        return EFAULT;
    }

    memcpy(kbuf_dst, kbuf_src, len);
}
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location  
^X Exit ^R Read File ^\_ Replace ^U Paste ^J Justify ^\_ Go To Line

- ⇒ Procedez la fel, pun semnatura functiei insyscalls.masters si codul functiei in sys\_generic.c
- ⇒ Ma folosesc de malloc si free ca sa ocup loc in memorie pentru buffer ul copiat
- ⇒ Ma folosesc de copyin si copyout pentru a copia date din buffer in kernel si daca copierea returneaza un cod diferit de 0 arunc eroare => invalid memory address
- ⇒ Recompiliez kernel ul si dau reboot





```
OpenBSD7.4 [Running] - Oracle VirtualBox
Bytes copiat: 7
Buff dest: copiat
labS0# cat test_functie2.c
#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>
#include <string.h>

int main() {
    char src[20] = "copiat";
    char dst[20] = {0};

    int copied = syscall(332, src, dst, strlen(src) + 1);

    printf("Bytes copiat: %d\n", copied);
    printf("Buff dest: %s\n", dst);

    return 0;
}

labS0# ./test_functie2
Bytes copiat: 7
Buff dest: copiat
⇒ labS0# $
```

- ⇒ Apelez syscall ul din kernel cu un program in c si observ ca la adresa vectorului de chars dest initializat cu 0 uri au fost mutate bitii din string ul src