Υποβολή 2ης Άσκησης

Χαραλαμπόπουλος Παναγιωτης	5681

Ερώτημα 1

Λίστα τροποποιηθέντων αρχείων

/usr/src/servers/vfs/open.c

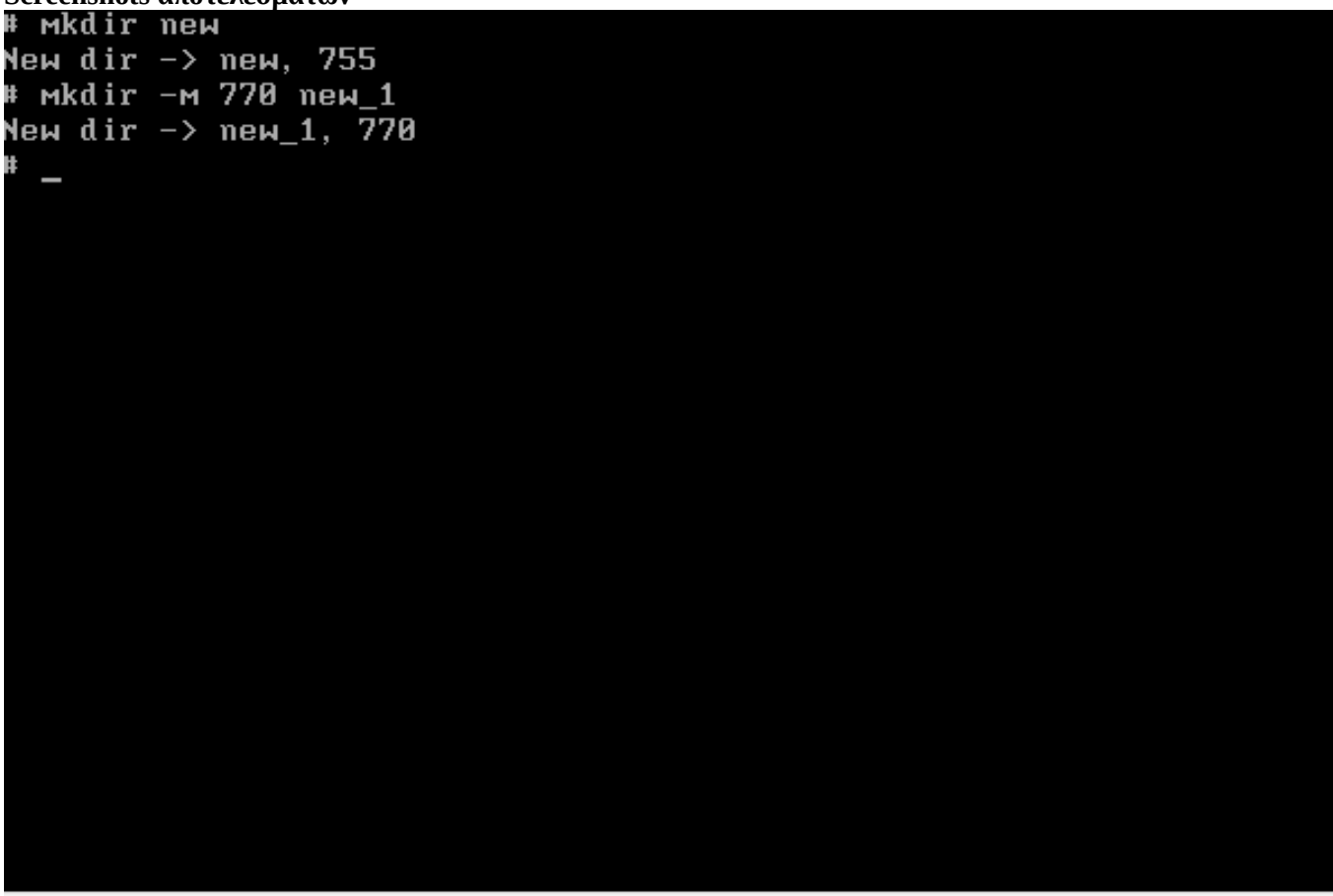
Τροποποιήσεις

/usr/src/servers/vfs/open.c

Line	Code
611	printf("New dir -> %s, %o",fullpath,dirmode);

```
607     } else if ((r = forbidden(fp, vp, W_BIT|X_BIT)) == OK) {
608       r = req_mkdir(vp->v_fs_e, vp->v_inode_nr, fullpath, fp->fp_effuid,
609                     fp->fp_effgid, bits);
610     }
611     printf("New dir -> %s, %o\n",fullpath,dirmode);
612     unlock_vnode(vp);
613     unlock_vmnt(vmp);
614     put_vnode(vp);
615     return(r);
616   }
```

Screenshots αποτελεσμάτων



Ερώτημα 2

Λίστα τροποποιηθέντων αρχείων

/usr/src/lib/libsys/sys_fork.c

Τροποποιήσεις

/usr/src/lib/libsys/sys_fork.c

Line	Code
21	printf("process scheduled!\n");

```
12    message m;
13    int r;
14
15    m.PR_ENDPT = parent;
16    m.PR_SLOT = child;
17    m.PR_FORK_FLAGS = flags;
18    r = _kernel_call(SYS_FORK, &m);
19    *child_endpoint = m.PR_ENDPT;
20    *msgaddr = (vir_bytes) m.PR_FORK_MSGADDR;
21    printf("process scheduled!\n");
22    return r;
23  }
```

Σημ:Για το ερωτημα χρησιμοποιηθηκε το αρχειο(βοηθητική συνάρτηση) sys_fork.c που εκτελει το kernel call του SYS_FORK μεσω της kernel_call .Το ιδιο αποτελεσμα θα μπορουσα να εχω και με την εξης τροποποιηση του handler function που αντιστοιχιζεται στο SYS_FORK:

/usr/src/kernel/system/do_fork.c

Line	Code
125	printf("process scheduled!\n");

Screenshots αποτελεσμάτων

Σημ: Για τον καλυτερο ελεγχο των αποτελεσματων δημιουργηθηκε και ενα αρχειο test.c το οποιο μεταγλωτιστηκε με την εντολη clang –o testFork test.c και παραχθηκε το εκτελεσιμο testFork το οποιο καλουμε με την ./testFork .Ο κωδικας που περιεχει το test.c ειναι ο παρακατω:

```
#include <unistd.h>
#include <stdio.h>
int main(int argc, char** args) {
pid_t id = fork();
printf("Process ID: %d\n", id);
return 0;
}
```

```
# ls
process scheduled!
bin           dev           libexec       sbin          tmp
boot          etc           mnt           sys           usr
boot.cfg      home          proc          test.c        var
boot_monitor  lib           root          testFork
# bash
process scheduled!
bash-4.2# ./testFork
process scheduled!
process scheduled!
Process ID: 158
Process ID: 0
bash-4.2# _
```

Ερώτημα 3

Λίστα τροποποιηθέντων αρχείων

/usr/src/include/minix/callnr.h
/usr/src/servers/vfs/table.c
/usr/src/servers/vfs/proto.h
/usr/src/servers/vfs/open.c
/usr/src/lib/libc/sys-minix/mkdir.c
/usr/src/bin/mkdir/mkdir.c
/usr/src/sys/sys/stat.h

Τροποποιήσεις

/usr/src/include/minix/callnr.h

Line	Code
68	#define MYMKDIR 69

```
66    #define GETMCONTEXT        67
67    #define SETMCONTEXT        68
68    #define MYMKDIR   69
69    /* Posix signal handling. */
70    #define SIGACTION     71
```

/usr/src/servers/vfs/table.c

Line	Code
87	do_mymkdir, /* 69 = mymkdir */

```
85      do_lstat,   /* 67 = lstat - badly numbered, being phased out */
86      no_sys,     /* 68 = unused  */
87      do_mymkdir,     /* 69 = mymkdir  */
88      no_sys,     /* 70 = unused  */
89      no_sys,     /* 71 = (sigaction) */
```

/usr/src/servers/vfs/proto.h

Line	Code
173	int do_mymkdir(void);

```
171 int do_vm_open(void);
172 int do_vm_close(void);
173 int do_mymkdir(void);
174 /* path.c */
```

/usr/src/servers/vfs/open.c

Line	Code
794	/*===*
795	* do_mymkdir *
796	*===*/
797	int do_mymkdir()
798	{
799	/* Perform the mkdir(name, mode) system call. */
800	mode_t bits; /* mode bits for the new inode */
801	int r;
802	struct vnode *vp;
803	struct vmnt *vmp;
804	char fullpath[PATH_MAX];
805	struct lookup resolve;
806	vir_bytes vname1;
807	size_t vname1_length;
808	mode_t dirmode;
809	
810	vname1 = (vir_bytes) job_m_in.name1;
811	vname1_length = (size_t) job_m_in.name1_length;
812	dirmode = (mode_t) job_m_in.mode;
813	
814	lookup_init(&resolve, fullpath, PATH_NOFLAGS, &vmp, &vp);
815	resolve.l_vmnt_lock = VMNT_WRITE;
816	resolve.l_vnode_lock = VNODE_WRITE;
817	

```
818 if (fetch_name(vname1, vname1_length, fullpath) != OK) return(err_code);
819 bits = I_DIRECTORY | (dirmode & RWX_MODES & fp->fp_umask);
820 if ((vp = last_dir(&resolve, fp)) == NULL) return(err_code);
821
822 /* Make sure that the object is a directory */
823 if (!S_ISDIR(vp->v_mode)) {
824 r = ENOTDIR;
825 } else if ((r = forbidden(fp, vp, W_BIT|X_BIT)) == OK) {
826 r = req_mkdir(vp->v_fs_e, vp->v_inode_nr, fullpath, fp->fp_effuid,
827 fp->fp_effgid, bits);
828 }
829 unlock_vnode(vp);
830 unlock_vmnt(vmp);
831 put_vnode(vp);
832 return(r);
833 }
```

```
794 /*===========================================================================*
795  *              do_mymkdir                   *
796  *===========================================================================*/
797 int do_mymkdir()
798 {
799 /* Perform the mkdir(name, mode) system call. */
800   mode_t bits;          /* mode bits for the new inode */
801   int r;
802   struct vnode *vp;
803   struct vmnt *vmp;
804   char fullpath[PATH_MAX];
805   struct lookup resolve;
806   vir_bytes vname1;
807   size_t vname1_length;
808   mode_t dirmode;
809
810   vname1 = (vir_bytes) job_m_in.name1;
811   vname1_length = (size_t) job_m_in.name1_length;
812   dirmode = (mode_t) job_m_in.mode;
813
814   lookup_init(&resolve, fullpath, PATH_NOFLAGS, &vmp, &vp);
815   resolve.l_vmnt_lock = VMNT_WRITE;
816   resolve.l_vnode_lock = VNODE_WRITE;
817
818   if (fetch_name(vname1, vname1_length, fullpath) != OK) return(err_code);
819   bits = I_DIRECTORY | (dirmode & RWX_MODES & fp->fp_umask);
820   if ((vp = last_dir(&resolve, fp)) == NULL) return(err_code);
821
822   /* Make sure that the object is a directory */
823   if (!S_ISDIR(vp->v_mode)) {
824     r = ENOTDIR;
825   } else if ((r = forbidden(fp, vp, W_BIT|X_BIT)) == OK) {
826     r = req_mkdir(vp->v_fs_e, vp->v_inode_nr, fullpath, fp->fp_effuid,
827                   fp->fp_effgid, bits);
828   }
829   unlock_vnode(vp);
830   unlock_vmnt(vmp);
831   put_vnode(vp);
832   return(r);
833 }
```

/usr/src/lib/libc/sys-minix/mkdir.c

Line	Code
22	int mymkdir(const char *name, mode_t mode)
23	{
24	message m;
25	
26	m.m1_i1 = strlen(name) + 1;
27	m.m1_i2 = mode;
28	m.m1_p1 = (char *) __UNCONST(name);
29	return(_syscall(VFS_PROC_NR, MYMKDIR, &m));
30	}

```
22  int mymkdir(const char *name, mode_t mode)
23  {
24    message m;
25
26    m.m1_i1 = strlen(name) + 1;
27    m.m1_i2 = mode;
28    m.m1_p1 = (char *) __UNCONST(name);
29    return(_syscall(VFS_PROC_NR, MYMKDIR, &m));
30  }
```

/usr/src/bin/mkdir/mkdir.c

Line	Code
67	char cwd[1024];
123	if (mymkdir(*argv, mode) < 0) {
138	else {
139	if (getcwd(cwd, sizeof(cwd)) != NULL) {
140	printf("\nFull path to new dir: %s", cwd);
141	if (strcmp(cwd,"/") != 0) {
142	printf("/");
143	}
144	printf("%s , %o",*argv,mode);
145	printf("\n\n");
146	}
147	
148	
149	
150	}

184	rv = mymkdir(path, done ? mode : dir_mode);

```
67      char cwd[1024];
123         if (mymkdir(*argv, mode) < 0) {
119       if (pflag) {
120           if (mkpath(*argv, mode, dir_mode) < 0)
121               exitval = EXIT_FAILURE;
122       } else {
123           if (mymkdir(*argv, mode) < 0) {
124               warn("%s", *argv);
125               exitval = EXIT_FAILURE;
126           } else {
127               /*
128                * The mkdir() and umask() calls both honor
129                * only the file permission bits, so if you try
130                * to set a mode including the sticky, setuid,
131                * setgid bits you lose them. So chmod().
132                */
133               if ((mode & ~(S_IRWXU|S_IRWXG|S_IRWXO)) != 0 &&
134                   chmod(*argv, mode) == -1) {
135                   warn("%s", *argv);
136                   exitval = EXIT_FAILURE;
137               }
138               else {
139                   if (getcwd(cwd, sizeof(cwd)) != NULL) {
140                       printf("\nFull path to new dir: %s", cwd);
141                       if ( strcmp(cwd,"/") != 0 ) {
142                           printf("/");
143                       }
144                       printf("%s , %o",*argv,mode);
145                       printf("\n\n");
146                   }
147
148
149
150               }
151
152
153
154           }
155       }
184         rv = mymkdir(path, done ? mode : dir_mode);
```

/usr/src/sys/sys/stat.h

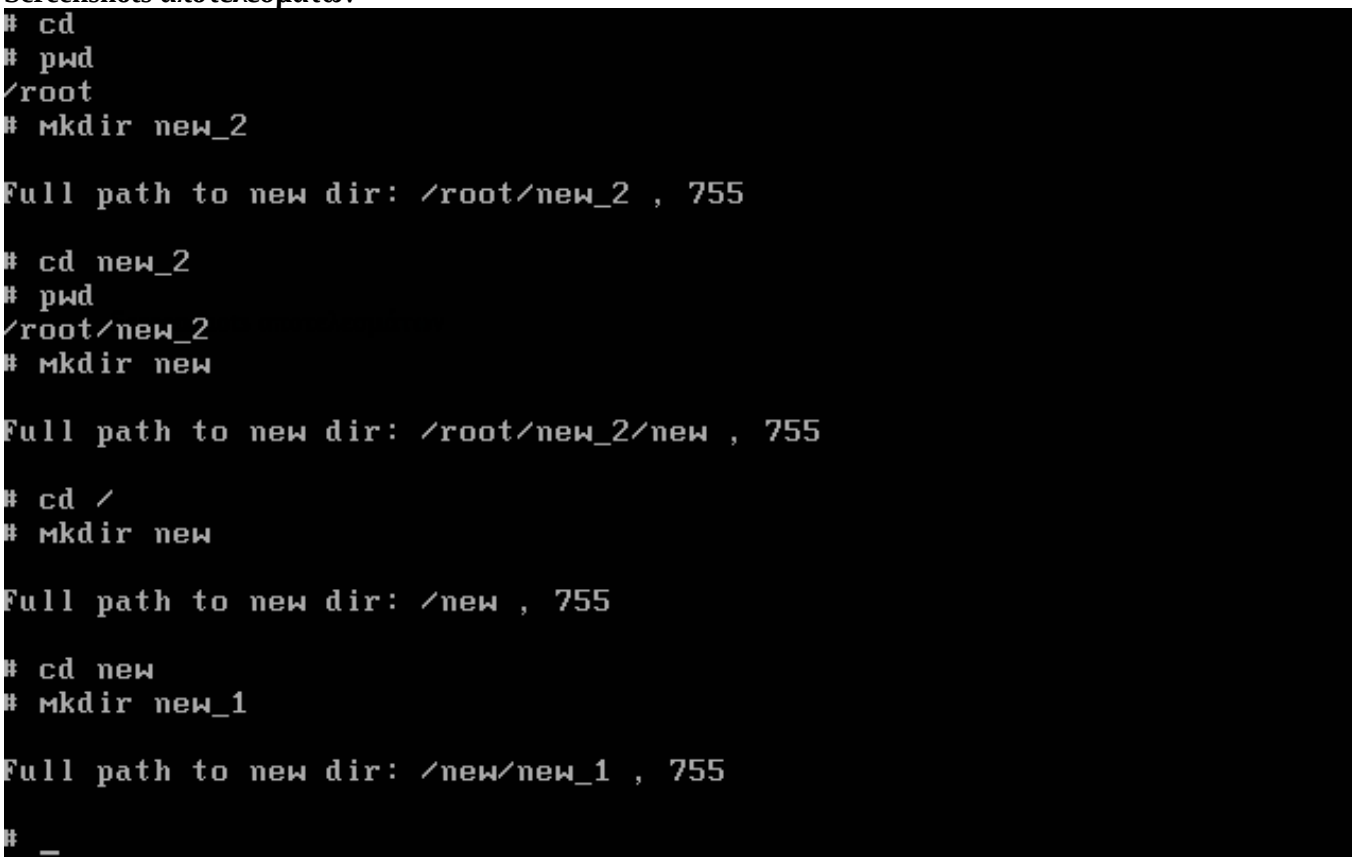
Line	Code
229	int mymkdir(const char *, mode_t);

```
229 int mymkdir(const char *, mode_t);
```

Screenshots αποτελεσμάτων

Στην επομενη σελιδα:

Screenshots αποτελεσμάτων



Ερώτημα 4

Λίστα τροποποιηθέντων αρχείων

/usr/src/include/minix/callnr.h
/usr/src/servers/pm/table.c
/usr/src/servers/pm/proto.h
/usr/src/servers/pm/Makefile
/usr/src/servers/pm/my_syscall.c

Τροποποιήσεις

/usr/src/include/minix/callnr.h

Line	Code
69	#define MYSYSCALL 70

/usr/src/servers/pm/table.c

Line	Code
84	do_mysyscall, /* 70 = mysyscall*/

/usr/src/servers/pm/proto.h

Line	Code
10	/* my_syscall.c */
11	int do_mysyscall(void);

/usr/src/servers/pm/Makefile

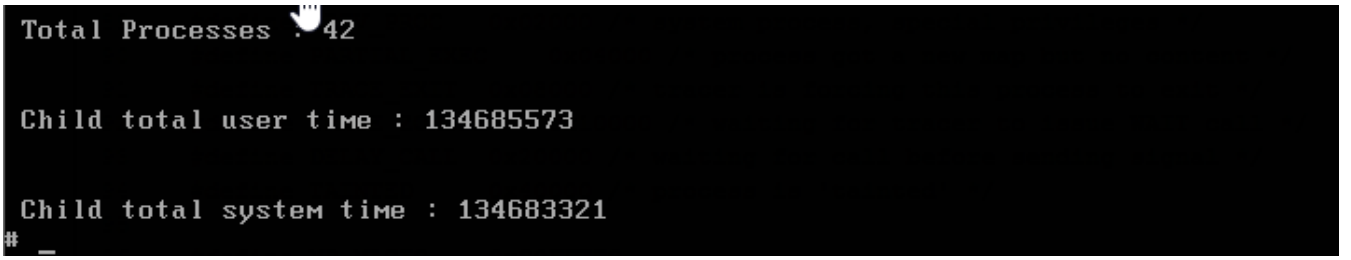
Line	Code
5	SRCS= main.c forkexit.c break.c exec.c time.c alarm.c my_syscall.c\

/usr/src/servers/pm/my_syscall.c

Line	Code
1	#include "pm.h"
2	#include <sys/wait.h>
3	#include <assert.h>
4	#include <minix/callnr.h>
5	#include <minix/com.h>
6	#include <minix/sched.h>
7	#include <minix/vm.h>
8	#include <sys/ptrace.h>
9	#include <sys/resource.h>
10	#include <signal.h>
11	#include "mproc.h"
12	#include "param.h"
13	
14	
15	int do_mysyscall()
16	{
17	int cnt_2,cnt,totalutime,totalstime;
18	for (cnt = 0; cnt<NR_PROCS; cnt++) {
19	if (mproc[cnt].mp_flags & IN_USE){
20	cnt_2++;
21	totalutime += mproc[cnt].mp_child_utime;
22	totalstime += mproc[cnt].mp_child_stime;
23	}
24	}

25	printf("\n\n Total Processes : %d\n",cnt_2);
26	printf("\n\n Child total user time : %d\n",totalutime);
27	printf("\n\n Child total system time : %d\n",totalstime);
28	return 0;
29	}

Screenshots αποτελεσμάτων



Ερώτημα 5

Λίστα τροποποιηθέντων αρχείων

/usr/src/include/minix/callnr.h
/usr/src/servers/pm/table.c
/usr/src/servers/pm/proto.h
/usr/src/servers/pm/my_syscall.c
/usr/src/lib/libc/sys-minix/my_libcall.c
/usr/src/lib/libc/sys-minix/Makefile.inc

Τροποποιήσεις

/usr/src/include/minix/callnr.h

Line	Code
57	#define MYLIBCALL 56

/usr/src/servers/pm/table.c

Line	Code
70	do_mylibcall, /* 70 = mylibcall*/

/usr/src/servers/pm/proto.h

Line	Code
12	int do_mylibcall(void);