ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΑΣΚΗΣΗ 4

ΟΝΟΜΑ: ΣΑΒΒΑΣ ΓΕΩΡΓΙΟΥ	ΑΜ : 235037 / 1040803

ΕΡΩΤΗΜΑ Α

Λίστα τροποποιηθέντων αρχείων

- /usr/src/servers/vfs/protect.c

Τροποποιήσεις

/usr/src/servers/vfs/protect.c

Line	Code
95	printf("chmod: %s %o\n",fullpath,new_mode);

ΕΡΩΤΗΜΑ_Β

Λίστα τροποποιηθέντων αρχείων

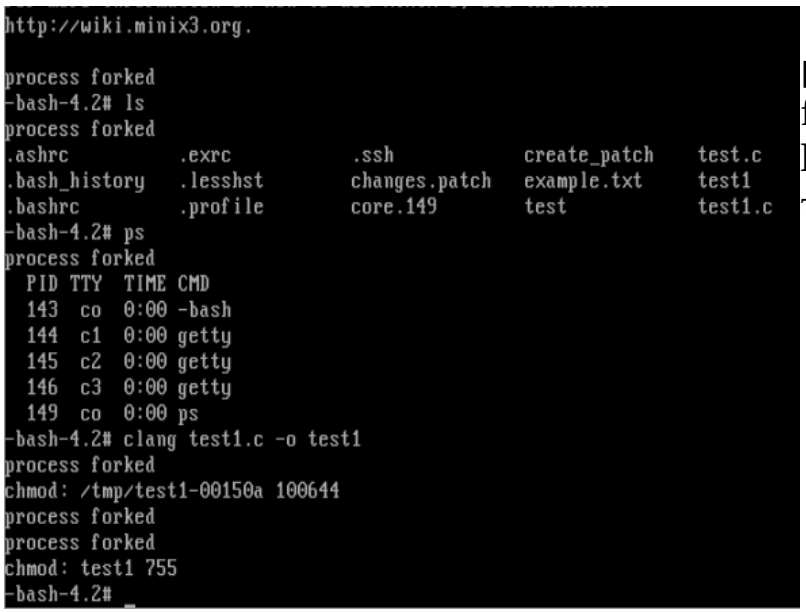
- /usr/src/kernel/system/do_fork.c

Τροποποιήσεις

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

Line	Code
110	printf("process forked\n");

Κατω απο το " RTS_SET(rpc , RTS_VMINHIBIT); " οπου ειναι η ενημερωση του process table του kernel που βρισκονται στο /usr/src/kernel/proc.h οι δηλωσεις.



Οπως φαινεται στην εικονα , μια νεα διεργασια γινεται forked μετα απο το login , ενα ls , το ps και ενα compile με το clang.

ΕΡΩΤΗΜΑ_Γ

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/misc.c
- /usr/include/mycalls.h

Τροποποιήσεις

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

Line	Code
68	#define PROCNUM 69

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

Line	Code
83	do_procnum, /* 69 = number of running processes*/

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

Line	Code
59	int do_procnum(void);

/usr/src/servers/pm/misc.c

Line	Code
32	#include "glo.h"
67-76	/ *========== ==========* * do_procnum * *========== ==========*/ int do_procnum(){

	int numberofproc = procs_in_use;
	//procs_in_use a variable in glo.h
	return numberofproc;
	}

/usr/include/mycalls.h

Line	Code

1-8	#include <lib.h>
	#include <unistd.h>
	
	int procnum(){
	
	message m;
	
	return(_syscall(PM_PROC_NR,PROCNUM,
	&m));
	}

Εφτιαξα ενα προγραμμα για τεστ του system call 69(procnum)

test69.c

```
#include <stdio.h>
#include <sys/types.h>
#include <mycalls.h>
#include <sys/stat.h>
#include <stdlib.h>

int main(int argc , char *argv[]){
        int numofproc = 0;
        numofproc = procnum();
        printf("%d\n" , numofproc);
}
```

Το process forked ειναι απο το προηγουμενο ερωτημα.Το τυπωμα στην οθονη ειναι απο το test69.c οπως φαινεται πιο πανω και οχι απο το system call.

ΕΡΩΤΗΜΑ Δ

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/misc.c
- /usr/include/mycalls.h

Τροποποιήσεις

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

Line	Code
69	#define FATHERPID 70

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

Line	Code
84	do_fatherpid, /* 70 = returns the pid of the father process*/

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

Line	Code
60	int do_fatherpid(int childpid);

/usr/src/servers/pm/misc.c

Line	Code
32	#include "glo.h"
78-95	/ *========== ========== ====* * do_fatherpid * *========== ========== =====*/ int do_fatherpid(){

	int givenpid = m_in.m1_i1;
	int fathprocpid = 0;
	int i , fathindex;
	for(i=0; i<NR_PROCS; i++){
	if(givenpid == mproc[i].mp_pid){
	fathindex = mproc[i].mp_parent;
	fathprocpid = mproc[fathindex].mp_pid;
	break;
	}
	}
	return fathprocpid;
	}

/usr/include/mycalls.h

Line	Code

9-15	int fatherpid(int childpid){
	message m;
	m.m1_i1 = childpid;
	
	return(_syscall(PM_PROC_NR,FATHERPID,&m));
	
	}

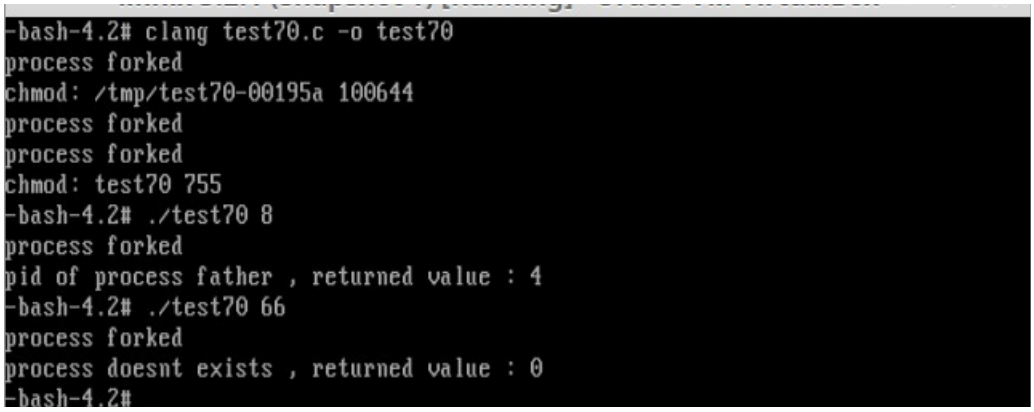
Εφτιαξα ενα προγραμμα για τεστ του system call 70(fatherpid)

test70.c

```
#include <stdio.h>
#include <sys/types.h>
#include <mycalls.h>
#include <sys/stat.h>
#include <stdlib.h>

int main(int argc , char *argv[]){
        if(argc==2){
                int givenpid = atoi(argv[1]);
                int ans;
                ans = fatherpid(givenpid);
```

```
            if(ans == 0 ){
                  printf("process doesnt exists , returned value : %d \n" , ans);
            }else{
                  printf("pid of process father , returned value : %d \n", ans);
            }
      }
}
```



Τα process forked οπως και το chmod ειναι απο τα προηγουμενα ερωτηματα.Το τυπωμα στην οθονη ειναι απο το test70.c οπως φαινεται πιο πανω και οχι απο το system call.

Η τιμη που επεστρεψε για το pid 8 ειναι το pid 4 , ενω για το 66 αφου δεν υπηρχε σαν pid , επεστρεψε 0.