1η Εργαστηριακή Άσκηση (Λύση)

Στη συνέχεια παρατίθενται τα τμήματα κώδικα που πρέπει να προστεθούν ή να μεταβληθούν στην υπάρχουσα υλοποίηση του File Server έτσι ώστε να υποστηριχτεί η system-call `getprocs()`.

/usr/include/minix/callnr.h

callnr.h:1
```
-#define NCALLS 77
+#define NCALLS 78
```

callnr.h:66
```
#define GETPROCS 77
```

Αλλάζουμε το πλήθος των system-calls και προσθέτουμε ένα id (77) με macro name `GETPROCS` για την `getprocs()`.

/usr/src/fs/table.c

table.c:96
```
+do_getprocs,      /* 77=GETPROCS */
```

Αλλάζουμε το `call_vec[]` του FS δηλώνοντας ως συνάρτηση εξυπηρέτησης του μηνύματος με `type GETPROCS` την `do_getprocs()`.

/usr/src/mm/table.c

table.c:94
```
+no_sys,           /* 77=GETPROCS */
```

Το `call_vec[]` του MM πρέπει να έχει μήκος `CALL_NR`. Για αυτόν το λόγο δηλώνουμε στη θέση `77` του `call_vec[]` ως συνάρτηση εξυπηρέτησης του μηνύματος με `type GETPROCS` την `no_sys()`, η οποία απλά επιστρέφει το λάθος `EINVAL` δηλώνοντας ότι ο MM δεν διαχειρίζεται το μήνυμα με `type GETPROCS`.

/usr/src/fs/fproc.h

fproc.h:33
```
+   char fp_inuse;
```

Το μέλος `fp_inuse` που προστίθεται στη δομή `fproc` παίρνει τις τιμές `TRUE` ή `FALSE`. Η τιμή `TRUE` δείχνει ότι η συγκεκριμένη θέση του πίνακα `fproc[]`, που διατηρεί ο FS, αντιστοιχεί σε μία υπάρχουσα διεργασία ενώ η τιμή `FALSE` το αντίθετο.

/usr/src/fs/misc.c

misc.c:264

```
+    fp->fp_inuse = FALSE;
```

misc.c:341

```
+ PUBLIC int do_getprocs(void)
+ {
+      int r;
+      int fi;
+      int i;
+      ino_t i_num;
+      struct fproc *fp;
+      struct inode *ind;
+      pid_t res_pids[NR_PROCS - LOW_USER];
+
+    if (!super_user) return (EPERM);
+      if ((r = fetch_name(c_name, name1_length, M1)) != OK) return (err_code);
+      /* printf("File: %s\n", user_path); */
+      if ((ind = eat_path(user_path)) == NIL_INODE) return (err_code);
+      i_num = ind->i_num;
+      /*printf("Inode: %u\n", ind->i_num);*/
+
+      fi = 0;
+    for (fp = &fproc[LOW_USER]; fp < &fproc[NR_PROCS] && fi < m.m1_i2; fp++) {
+          if (fp->fp_inuse == TRUE) {
+                /* printf("%d :", fp->fp_pid); */
+                for (i = 0; i < OPEN_MAX; i++) {
+                      if (fp->fp_filp[i] == NIL_FILP) continue;
+                      if (fp->fp_filp[i]->filp_mode == FILP_CLOSED) continue;
+                      /*printf("%u :", fp->fp_filp[i]->filp_ino->i_num); */
+                      if (fp->fp_filp[i]->filp_ino->i_num == i_num) {
+                          res_pids[fi++] = fp->fp_pid;
+                          /* printf(" |%d| ", fp->fp_pid); */
+                          break;
+                      }
+                }
+          }
+      }
+    r = sys_copy(FS_PROC_NR, D, (phys_bytes) res_pids,\
+                  who, D, (phys_bytes) m.m1_p2, (phys_bytes) fi * sizeof(int));
+    return fi;
+ }
+
```

Η πρώτη αλλαγή που πρέπει να γίνει στο αρχείο */usr/src/fs/misc.c* αφορά την συνάρτηση `do_exit()` η οποία εκτελείται από τον FS για να εξυπηρετήσει το μήνυμα με `type EXIT` που του αποστέλλει ο MM (όταν αυτός με την σειρά του λάβει από κάποια user-level διαδικασία αίτηση εξυπηρέτησης για την system-call `exit()`). Συγκεκριμένα ο κώδικας που προστίθεται, απλά θέτει στο μέλος `fp_inuse` της συγκεκριμένης struct την τιμή `FALSE`, για να υποδείξει ότι πλέον δεν αντιστοιχεί σε υπάρχουσα διεργασία.
Η δεύτερη αλλαγή αποτελεί τον ορισμό της συνάρτησης `do_getprocs()` που εξυπηρετεί την system-call `getprocs()`.

/usr/src/fs/main.c

main.c:155

```
+       fp->fp_inuse = TRUE;
    }
+   for (i = LOW_USER + 1; i < NR_PROCS; i++) {
+           fp = &fproc[i];
+           fp->fp_inuse = FALSE;
+   }
```

Η προσθήκη αυτή γίνεται στην συνάρτηση `fs_init()` η οποία μεταξύ άλλων αρχικοποιεί και τις δομές του πίνακα `fproc[]`.

getprocs.h

```
+#ifndef _GETPROCS_H_
+#define _GETPROCS_H_
+#include <lib.h>
+#define getprocs _getprocs
+#include <unistd.h>
+#include <string.h>

+int getprocs(const char *_name, const int _size, int *_procs, const int _n)
+{
+           message m;
+
+           m.m1_p1 = (char *) _name;
+           m.m1_p2 = (char *) _procs;
+           m.m1_i1 = (int) _size;
+           m.m1_i2 = (int) _n;
+           return (_syscall(FS, GETPROCS, &m));
+}
+#endif
```

Το header file *getprocs.h* πρέπει να συμπεριληφθεί (`#include "getprocs.h"`) στα αρχεία που χρησιμοποιούν την `getprocs()`. Ο χρήστης πρέπει να έχει το αρχείο *getprocs.h* στον ίδιο κατάλογο αρχείων με αυτόν του κώδικα που καλεί την `getprocs()`.