

Ιωάννης Λεοντιάδης (1046836)

Χρήστος Φείδας

Λειτουργικά Συστήματα (ECE_ΓΚ702)

Οκτώβριος 2023

Δραστηριότητα 1: Εισαγωγή στο xv6 Kernel & Προσθήκη System Call

Το λογισμικό που αναπτύσσεται στο πλαίσιο των δραστηριοτήτων και πληροφορίες για τα εργαλεία που χρησιμοποιούνται βρίσκονται στο ηλεκτρονικό αποθετήριο [os-activities](#). Για να αποκτήσετε δικαίωμα πρόσβασης παρακαλώ όπως στείλετε τα στοιχεία του GitHub λογαριασμού σας (όνομα χρήστη ή διεύθυνση ηλεκτρονικού ταχυδρομείου). Κάθε δραστηριότητα αναπτύσσεται σε ξεχωριστό κλάδο. Η παρούσα δραστηριότητα βρίσκεται στον φάκελο [xv6](#).

Άσκηση 1

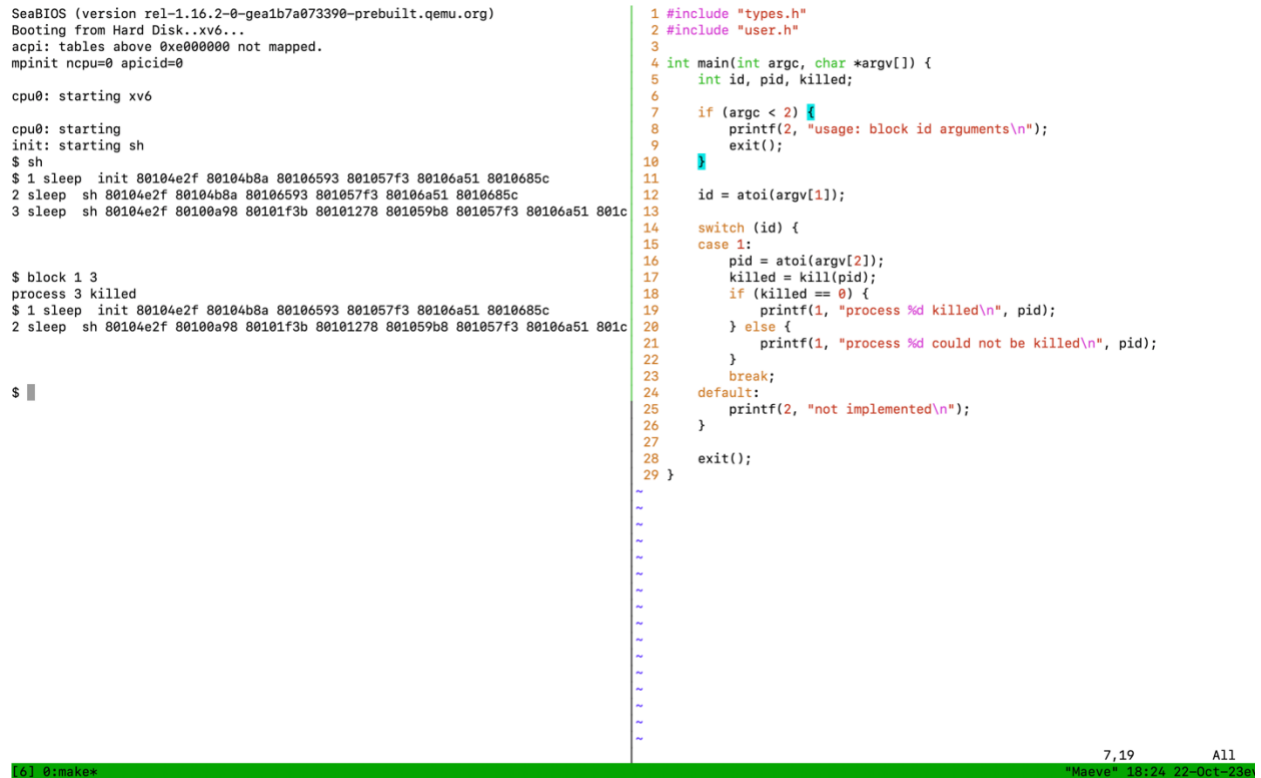
Στο αρχείο `include/user.h` βρίσκονται οι ορισμοί των κλήσεων συστήματος.

Το πρόγραμμα `kill.c` ζητάει από το λειτουργικό σύστημα να τερματίσει ένα σύνολο διεργασιών. Λαμβάνει ως ορίσματα τα αναγνωριστικά των διεργασιών (PID) και χρησιμοποιώντας την κλήση συστήματος `kill` καλεί το λειτουργικό σύστημα να τις τερματίσει. Στην περίπτωση που δεν έχουν δοθεί ορίσματα, χρησιμοποιείται η συνάρτηση `printf` για να τυπωθεί ένα μήνυμα σφάλματος. Η `printf` μεταφέρει κάθε χαρακτήρα στη βασική ροή σφαλμάτων χρησιμοποιώντας την κλήση συστήματος `write`. Για να τερματίσει το πρόγραμμα, καλεί την κλήση συστήματος `exit`.

Άσκηση 2

Το πρόγραμμα block εκτελεί ένα μπλοκ πηγαίου κώδικα ανάλογα με το αναγνωριστικό που δίνεται ως όρισμα. Χρησιμοποιεί τη δήλωση switch για την επιλογή του κατάλληλου μπλοκ.

Το μπλοκ της περίπτωσης 1 καλεί τη συνάρτηση συστήματος kill για να τερματίσει μια διεργασία (Εικόνα 1).



```
SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org)
Booting from Hard Disk..xv6...
acpi: tables above 0xe000000 not mapped.
mpinit ncpu=0 apicid=0

cpu0: starting xv6

cpu0: starting
init: starting sh
$ sh
$ 1 sleep init 80104e2f 80104b8a 80106593 801057f3 80106a51 8010685c
2 sleep sh 80104e2f 80104b8a 80106593 801057f3 80106a51 8010685c
3 sleep sh 80104e2f 80100a98 80101f3b 80101278 801059b8 801057f3 80106a51 8010c

$ block 1 3
process 3 killed
$ 1 sleep init 80104e2f 80104b8a 80106593 801057f3 80106a51 8010685c
2 sleep sh 80104e2f 80100a98 80101f3b 80101278 801059b8 801057f3 80106a51 8010c

$ █
```

```
1 #include "types.h"
2 #include "user.h"
3
4 int main(int argc, char *argv[]) {
5     int id, pid, killed;
6
7     if (argc < 2) {
8         printf(2, "usage: block id arguments\n");
9         exit();
10    }
11
12    id = atoi(argv[1]);
13
14    switch (id) {
15    case 1:
16        pid = atoi(argv[2]);
17        killed = kill(pid);
18        if (killed == 0) {
19            printf(1, "process %d killed\n", pid);
20        } else {
21            printf(1, "process %d could not be killed\n", pid);
22        }
23        break;
24    default:
25        printf(2, "not implemented\n");
26    }
27
28    exit();
29 }
```

Εικόνα 1. Το πρόγραμμα χρήστη block και χρήση του για τον τερματισμό μιας διεργασίας

Άσκηση 3

Η συνάρτηση συστήματος `sys_uptime` αναφέρεται σε τρία σημεία του πηγαίου κώδικα του πυρήνα (Εικόνα 2).

```
→ xv6 git:(activity-1) ✕ grep -nrw 'sys_uptime' kernel
kernel/sysproc.c:93:int sys_uptime(void) {
kernel/syscall.c:129:extern int sys_uptime(void);
kernel/syscall.c:143:    [SYS_sleep] sys_sleep,          [SYS_uptime] sys_uptime,
→ xv6 git:(activity-1) ✕
```



Εικόνα 2. Αναζήτηση της συνάρτησης συστήματος `sys_uptime` στον πυρήνα

Η `sys_uptime` επιστρέφει τον αριθμό χρονικών μονάδων του συστήματος (ticks) που έχουν περάσει από την εκκίνηση του λειτουργικού συστήματος.

Άσκηση 4

Η συνάρτηση στον πυρήνα η οποία καλεί τη συνάρτηση συστήματος που αντιστοιχεί σε κάποια κλήση συστήματος είναι η `syscall.c:syscall`.

Ο πηγαίος κώδικας που αφορά τις κλήσεις συστήματος και τον χειρισμό τους από το λειτουργικό σύστημα βρίσκεται στα αρχεία που παρουσιάζονται στον Πίνακα 1.

Αρχείο	Λειτουργία	Περιγραφή
<code>include/syscall.h</code>	χρήστη και πυρήνα	Αντικατάσταση ονομάτων συναρτήσεων συστήματος με ακεραίους
<code>include/user.h</code>	χρήστη	Ορισμοί διαθέσιμων συναρτήσεων συστήματος
<code>user/usys.S</code>	χρήστη	Ρουτίνες παγίδευσης προς τον πυρήνα
<code>kernel/vector.S</code>	πυρήνα	Ρουτίνες χειρισμού διακοπών
<code>kernel/trapasm.S</code>	πυρήνα	Δημιουργία πλαισίου παγίδευσης
<code>kernel/trap.c</code>	πυρήνα	Καθορισμός περιπτώσεων διακοπής
<code>kernel/syscall.c</code>	πυρήνα	Επιλογή συνάρτησης πυρήνα για συγκεκριμένη κλήση συστήματος

Πίνακας 1. Πηγαίος κώδικας χειρισμού κλήσεων συστήματος

Όταν πραγματοποιείται μια κλήση συστήματος εκτελείται η αντίστοιχη ρουτίνα παγίδευσης προς τον πυρήνα (`usys.S`). Η ρουτίνα παγίδευσης αποθηκεύει τον ακέραιο στον οποίο αντιστοιχεί η συνάρτηση συστήματος που ζητείται να εκτελεστεί και εκτελεί την εντολή `int $64` η οποία προκαλεί παγίδευση προς τον πυρήνα (Εικόνα 3). Ο επεξεργαστής αποθηκεύει την κατάσταση του, αλλάζει σε στοίβα και λειτουργία πυρήνα και εκτελεί τη ρουτίνα εξυπηρέτησης

διακοπής (ISR) που αντιστοιχεί σε διακοπή λόγω κλήσης συστήματος. Αυτή καθορίζεται από το στοιχείο 64 του πίνακα περιγραφής διακοπών (IDT) και είναι η `vectors.S:vector64`. Όπως όλες οι ρουτίνες εξυπηρέτησης διακοπών, έτσι και η `vector64` καλεί τη ρουτίνα `trapasm.S:alltraps` η οποία δημιουργεί το πλαίσιο παγίδευσης. Έπειτα καλείται η συνάρτηση `trap.c:trap` η οποία καθορίζει τον τύπο της διακοπής και τελικά καλεί την `syscall.c:syscall` ώστε να χειριστεί την διακοπή λόγω κλήσης συστήματος.

```
SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org)
Booting from Hard Disk..xv6...
acpi: tables above 0xe000000 not mapped.
mpinit ncpu=0 apicid=0

cpu0: starting xv6

cpu0: starting
init: starting sh
$ block 1

=> 0x97 <main+151>:      sub    $0xc,%esp
17      killed = kill(pid);
(gdb) s
=> 0x445 <kill>:        mov     $0x6,%eax
kill () at uilib/usys.S:18
18      SYSCALL(kill)
(gdb) s
=> 0x80107182 <vector64>:  push    $0x0
vector64 () at kernel/vectors.S:319
319     pushl $0
(gdb) n
=> 0x80107184 <vector64+2>:  push    $0x40
vector64 () at kernel/vectors.S:320
320     pushl $64
(gdb) n
=> 0x80107186 <vector64+4>:  jmp     0x80106913 <alltraps>
vector64 () at kernel/vectors.S:321
321     jmp alltraps
(gdb) s
=> 0x80106913 <alltraps>:    push    %ds
alltraps () at kernel/trapasm.S:7
7      pushl %ds
(gdb) n
=> 0x80106914 <alltraps+1>:  push    %es
alltraps () at kernel/trapasm.S:8
8      pushl %es
(gdb) bt
#0  alltraps () at kernel/trapasm.S:8
#1  0x0000002b in main (argc=2, argv=0x2fe8) at user/block.c:8
Backtrace stopped: previous frame inner to this frame (corrupt stack?)
(gdb) s
=> 0x80106915 <alltraps+2>:  push    %fs
alltraps () at kernel/trapasm.S:9
9      pushl %fs
(gdb) s
=> 0x80106917 <alltraps+4>:  push    %gs
alltraps () at kernel/trapasm.S:10
10     pushl %gs
(gdb) s
=> 0x80106919 <alltraps+6>:  pusha
alltraps () at kernel/trapasm.S:11
11     pushal
(gdb)
```

Εικόνα 3. Παγίδευση προς τον πυρήνα

Άσκηση 5

Η προσθήκη της κλήσης συστήματος `getfavnum` παρουσιάζεται στην Εικόνα 4.

```
SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org)
Booting from Hard Disk..xv6...
acpi: tables above 0xe000000 not mapped.
mpinit ncpu=0 apicid=0

cpu0: starting xv6

cpu0: starting
init: starting sh
$ block 2
my favorite number is 14
$
```

<pre>80 return -1; 81 } 82 sleep(&ticks, &tickslock); 83 } 84 release(&tickslock); 85 return 0; 86 } 87 88 // return how many clock tick interrupts have occurred 89 // since start. 90 int sys_uptime(void) { 91 uint xticks; 92 93 acquire(&tickslock); 94 xticks = ticks; 95 release(&tickslock); 96 return xticks; 97 } 98 99 int sys_getfavnum(void) { return 14; }</pre>	<pre>23 char *sbrk(int); 24 int sleep(int); 25 int uptime(void); 26 int getpinfo(struct pstat *); 27 int getfavnum(void); "include/user.h" 41 lines --60%-- 25,17 61% 19 #define SYS_unlink 18 20 #define SYS_link 19 21 #define SYS_mkdir 20 22 #define SYS_close 21 23 #define SYS_getpinfo 22 24 #define SYS_getfavnum 23 "include/syscall.h" 24 lines --91%-- 22,20 Bot 29 SYSCALL(sbrk) 30 SYSCALL(sleep) 31 SYSCALL(uptime) 32 SYSCALL(getpinfo) 33 SYSCALL(getfavnum) "ulib/usys.S" 33 lines --96%-- 32,17 Bot 134 [SYS_fork] sys_fork, [SYS_exit] sys_exit, 135 [SYS_wait] sys_wait, [SYS_pipe] sys_pipe, 136 [SYS_read] sys_read, [SYS_kill] sys_kill, 137 [SYS_exec] sys_exec, [SYS_fstat] sys_fstat, 138 [SYS_chdir] sys_chdir, [SYS_dup] sys_dup, 139 [SYS_getpid] sys_getpid, [SYS_sbrk] sys_sbrk, 140 [SYS_sleep] sys_sleep, [SYS_uptime] sys_uptime, 141 [SYS_open] sys_open, [SYS_write] sys_write, 142 [SYS_mknod] sys_mknod, [SYS_unlink] sys_unlink, 143 [SYS_link] sys_link, [SYS_mkdir] sys_mkdir, 144 [SYS_close] sys_close, [SYS_getpinfol] sys_getpinfol, 145 [SYS_getfavnum] sys_getfavnum, 146 }; 147 148 void syscall(void) { 149 int num; 150 151 num = proc->tf->eax; 152 if (num > 0 && num < NELEM(syscalls) && syscalls[num]) { 153 proc->tf->eax = syscalls[num](); "kernel/sysproc.c" 99 lines --100%-- 99,14 Bot "kernel/syscall.c" 158 lines --93%-- 148,16 96% "Maevs" 19:53 22-Oct-23</pre>
--	--

Εικόνα 4. Προσθήκη κλήσης συστήματος `getfavnum`

Άσκηση 6

Η προσθήκη της κλήσης συστήματος `halt` και η χρήση της μέσω του προγράμματος `shutdown` παρουσιάζεται στην Εικόνα 5.

SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org) Booting from Hard Disk..xv6... acpi: tables above 0xe000000 not mapped. mpinit ncpu=0 apicid=0					
cpu0: starting xv6					
cpu0: starting init: starting sh \$ shutdown → xv6 git:(activity-1) *					
	25 int uptime(void); 26 int getpinfo(struct pstat *); 27 int getfavnum(void); 28 int halt(void); 29				
	"include/user.h" 42 lines --64%--	27,17	64%		
	20 #define SYS_link 19 21 #define SYS_mkdir 20 22 #define SYS_close 21 23 #define SYS_getpinfo 22 24 #define SYS_getfavnum 23 25 #define SYS_halt 24				
	"include/syscall.h" 25 lines --88%--	22,16	Bot		
	29 SYSCALL(sbrk) 30 SYSCALL(sleep) 31 SYSCALL(uptime) 32 SYSCALL(getpinfo) 33 SYSCALL(getfavnum)				
	"ulib/usys.S" 34 lines --91%--	31,12	96%		
81 } 82 sleep(&ticks, &tickslock); 83 } 84 release(&tickslock); 85 return 0; 86 } 87 88 // return how many clock tick interrupts have occurred 89 // since start. 90 int sys_uptime(void) { 91 uint xticks; 92 93 acquire(&tickslock); 94 xticks = ticks; 95 release(&tickslock); 96 return xticks; 97 } 98 99 int sys_getfavnum(void) { return 14; } 100 int sys_halt(void) { outw(0x604, 0x2000); }	136 [SYS_wait] sys_wait, [SYS_pipe] sys_pipe, 137 [SYS_read] sys_read, [SYS_kill] sys_kill, 138 [SYS_exec] sys_exec, [SYS_fstat] sys_fstat, 139 [SYS_chdir] sys_chdir, [SYS_dup] sys_dup, 140 [SYS_getpid] sys_getpid, [SYS_sbrk] sys_sbrk, 141 [SYS_sleep] sys_sleep, [SYS_uptime] sys_uptime, 142 [SYS_open] sys_open, [SYS_write] sys_write, 143 [SYS_mknod] sys_mknod, [SYS_unlink] sys_unlink, 144 [SYS_link] sys_link, [SYS_mkdir] sys_mkdir, 145 [SYS_close] sys_close, [SYS_getpinfo] sys_getpinfo, 146 [SYS_getfavnum] sys_getfavnum, [SYS_halt] sys_halt, 147 };				
	"kernel/syscall.c" 159 lines --89%--	142,14	91%		
	1 #include "types.h" 2 #include "user.h" 3 4 int main(int argc, char *[]) { halt(); }				
	~				
"kernel/sysproc.c" 100 lines --100%--		100,43	Bot		
"user/shutdown.c" 4 lines --100%--		4,1	All		
[xv6] 0:vim*					"Maave" 20:02 22-Oct-23

Εικόνα 5. Προσθήκη κλήσης συστήματος `halt` και πρόγραμμα `shutdown`

Άσκηση 7

Η προσθήκη της κλήσης συστήματος getcount και ο έλεγχος της μέσω του προγράμματος block παρουσιάζονται στην Εικόνα 6.

```
SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org)
Booting from Hard Disk..xv6...
acpi: tables above 0xe000000 not mapped.
mpinit ncpu=0 apicid=0

cpu0: starting xv6

cpu0: starting
init: starting sh
$ break 3
exec break failed
$ block 3
getfavnum was called 0 times
calling getfavnum
getfavnum was called 1 times
calling getfavnum from another process
getfavnum was called 2 times
$

96     xticks = ticks;
97     release(&tickslock);
98     return xticks;
99 }
100
101 int sys_getfavnum(void) { return 14; }
102
103 int sys_halt(void) {
104     outw(0x604, 0x2000);
105     return 0;
106 }
107
108 int sys_getcount(void) {
109     int syscall;
110
111     if (argint(0, &syscall) < 0) {
112         return -1;
113     }
114     return syscalls_count[syscall];
115 }
"kernel/sysproc.c" 115 lines --88%--      102,0-1      Bot

27 int getfavnum(void);
28 int halt(void);
29 int getcount(int);
#include/user.h" 43 lines --65%--      28,15      65%

24 #define SYS_getfavnum 23
25 #define SYS_halt 24
26 #define SYS_getcount 25
#include/syscall.h" 26 lines --100%--      26,21      Bot

33 SYSCALL(getfavnum)
34 SYSCALL(halt)
35 SYSCALL(getcount)
ulib/usys.S" 35 lines --100%--      35,17      Bot

151 int syscalls_count[100];
152
153 void syscall(void) {
154     int num;
155
156     num = proc->tf->eax;
157     if (num > 0 && num < NELEM(syscalls) && syscalls[num]) {
158         syscalls_count[num]++;
159         proc->tf->eax = syscalls[num]();
160     }
"kernel/syscall.c" 164 lines --94%--      155,0-1      96%

27     case 3:
28         printf(1, "getfavnum was called %d times\n", getcount(23));
29         printf(1, "calling getfavnum\n");
30         getfavnum();
31         printf(1, "getfavnum was called %d times\n", getcount(23));
32         if (fork() == 0) {
33             printf(1, "calling getfavnum from another process\n");
34             getfavnum();
35         } else {
36             wait();
37             printf(1, "getfavnum was called %d times\n", getcount(23));
38         }
39         break;
40     default:
41         printf(2, "not implemented\n");
42     }
"user/block.c" 45 lines --82%--      37,17      89%

[xv6] 0:vim~
"Maave" 20:56 22-Oct-23
```

Εικόνα 6. Προσθήκη κλήσης συστήματος getcount και έλεγχος μέσω block

Άσκηση 8

Η προσθήκη της κλήσης συστήματος killrandom και ο έλεγχος της μέσω του προγράμματος παρουσιάζονται block στην Εικόνα 7.

\$ sleep init 80104e2f 80104b8a 801065cb 8010580a 80106b25 8010690	29 int getcount(int);			
2 sleep sh 80104e2f 80104b8a 801065cb 8010580a 80106b25 80106930	30 int killrandom(void);			
4 sleep sh 80104e2f 80104b8a 801065cb 8010580a 80106b25 80106930	31	"include/user.h" 44 lines --68%--	30,21	68%
5 sleep sh 80104e2f 80104b8a 801065cb 8010580a 80106b25 80106930	25 #define SYS_halt 24			
6 sleep sh 80104e2f 80100a98 80101f3b 80101278 801059cf 8010580a 0	26 #define SYS_getcount 25			
	27 #define SYS_killrandom 26			
	"include/syscall.h" 27L, 565B written		27,25	Bot
exec ailed	35 SYSCALL(getcount)			
\$ block 4	36 SYSCALL(killrandom)			
killed process 5	~			
\$ \$ block 4	"ulib/usys.S" 36 lines --100%--		36,14	Bot
killed process 4				
zombie!				
\$ 1 sleep init 80104e2f 80104b8a 801065cb 8010580a 80106b25 801060	145 [SYS_mknod] sys_mknod,	[SYS_unlink] sys_unlink,		
2 sleep sh 80104e2f 80100a98 80101f3b 80101278 801059cf 8010580a 0	146 [SYS_link] sys_link,	[SYS_mkdir] sys_mkdir,		
6 sleep sh 80104e2f 80100a98 80101f3b 80101278 801059cf 8010580a 0	147 [SYS_close] sys_close,	[SYS_getpinfo] sys_getpinfo,		
	148 [SYS_getfavnum] sys_getfavnum,	[SYS_halt] sys_halt,		
	149 [SYS_getcount] sys_getcount,	[SYS_killrandom] sys_killrandom,		
	150 };			
	151			
	152 int syscalls_count[100];			
	153			
	"kernel/syscall.c" 165 lines --90%--		149,16	92%
115 return syscalls_count[syscall];	34 getfavnum();			
116 }	35 } else {			
117	36 wait();			
118 int sys_killrandom(void) {	37 printf(1, "getfavnum was called %d times\n", getcount(23));			
119 int count = 0;	38 }			
120 int pid, check;	39 break;			
121	40 case 4:			
122 while(1){	41 int pid = killrandom();			
123 pid = random(NPROC);	42 printf(1, "killed process %d\n", pid);			
124 if (pid != 1 && pid != 0) {	43 break;			
125 check = kill(pid);	44 default:			
126 }	45 printf(2, "not implemented\n");			
127 if (check != -1){	46 }			
128 break;	47 exit();			
129 }	48 }			
130 }	49 }			
131 return pid;				
132 }				
133 }				
~				
"kernel/sysproc.c" 133 lines --96%--			128,18	Bot
[xv6] 0:vim*	"user/block.c" 49 lines --83%--		41,21	Bot
	"Maave" 22:18 22-Oct-20			

Εικόνα 7. Προσθήκη κλήσης συστήματος getcount και έλεγχος μέσω block

Για να βρεθούν τα αναγνωριστικά των διεργασιών που βρίσκονται σε κάποια από τις δυνατές καταστάσεις μπορεί να χρησιμοποιηθεί μια επανάληψη όπως αυτή της συνάρτησης προγραμματισμού εκτέλεσης των διεργασιών (scheduler).