Εργαστήριο Λειτουργικών Συστημάτων

Σεϊμένης Σπύρος	5070
Καλλιβωκάς Δημήτριος	4993

Άσκηση 3

1. Τροποποιήσεις Minix

Λίστα τροποποιηθέντων αρχείων Minix

/usr/src/servers/sched/schedule.c

Τροποποιήσεις αρχείων Minix

/usr/src/servers/sched/schedule.c

Line	**Code**
30	PUBLIC int do_noquantum(message *m_ptr)
:	:
34	unsigned long utime, stime;
35	struct proc pr;
:	:
:	:
43	rmp = $schedproc[proc_nr_n];
44	
45	sys_getproc(&pr, rmp->endpoint);
46	utime = pr.p_user_time;
47	stime = pr.p_sys_time;
48	if(utime>0 && stime/utime>1){
49	rmp->priority = 15;
50	}
51	
52	if (rmp->priority < MIN_USER_Q) {
:	:

2. Κώδικες δοκιμαστικών προγραμμάτων

cpu_bound.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/times.h>
#include <stdint.h>

static time_t real_start;
static time_t real_end;

static struct tms start_sys;
static struct tms end_sys;
static clock_t start;
static clock_t end;

void start_clock(void) {
        time(&real_start);
        start = times(&start_sys);
}

void end_clock(void) {
        time(&real_end);
        end = times(&end_sys);
}

void print_clock_results(void) {
        /* real, system, user */
        printf("%i, %d, %d \n",
                (int)(real_end - real_start),
    (intmax_t)(end_sys.tms_utime - start_sys.tms_utime),
                (intmax_t)(end_sys.tms_stime - start_sys.tms_stime));
}

int main(int arg, char **argv) {
    int i = 0;
    int j = 0;
    char *string;

    start_clock();

    if (arg>1)
        string = argv[1];
    else {
        printf("Usage: ./cpu_bound word\n");
        return 1;
    }

    while( i++ < 10000000 ) {
       for( j = 0; j < strlen(string)/2; j++ ) {
           if ( string[j] != string[strlen(string)-1-j])
               break;
       }

      /*if ( j == strlen(string)/2)
            printf("it is a palindrome\n");
      else
            printf("it is not a palindrome\n");
      */
    }

    end_clock();
    print_clock_results();
    return 0;
}
```

io_bound.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/times.h>
#include <stdint.h>

static time_t real_start;
static time_t real_end;

static struct tms start_sys;
static struct tms end_sys;
static clock_t start;
static clock_t end;

void start_clock(void) {
        time(&real_start);
        start = times(&start_sys);
}

void end_clock(void) {
        time(&real_end);
        end = times(&end_sys);
}

void print_clock_results(void) {
        /* real, user, system*/
        printf("%i, %d, %d \n",
        (int)(real_end - real_start),
        (intmax_t)(end_sys.tms_utime - start_sys.tms_utime),
        (intmax_t)(end_sys.tms_stime - start_sys.tms_stime));
}

int main(int arg, char **argv) {
    int i = 0;
    char buffer[100];
    FILE *fp;

    start_clock();

    if (arg<=1) {
        printf("Usage: ./io_bound filename\n");
        return 1;
    }

    while( i++ < 100000 ) {
        if (( fp = fopen(argv[1], "r") ) == NULL) {
            printf("File does not exist or couldn't be opened for reading
\n");
            return 1;
        }
        fread(buffer, 100, 1, fp);
        fseek(fp, 0, SEEK_SET);
        fclose(fp);
    }

    end_clock();
    print_clock_results();
    return 0;
}
```

bash script

```
####################################
# 5070 Seimenis Spyros
# 4993 Kallivokas Dimitris
####################################

#!/usr/bin/bash

rm -rf output
mkdir output
touch output/cpu_times.txt
touch output/io_times.txt

./cpu_bound racecar &
./io_bound cpu_bound.c &
wait

rm cpu_times.txt
rm io_times.txt
for((j=0 ; j < $1 ; j++))
do
        echo Loop$j --------- >> output/cpu_times.txt
        echo Loop$j --------- >> output/io_times.txt
        for i in 1 2 3
        do
                ./cpu_bound racecar    1>> output/cpu_times.txt&
                ./io_bound cpu_bound.c 1>> output/io_times.txt&
        done
        wait
done

echo "Testbench done!"
```

3. Screenshot εκτέλεσης προγραμμάτων

original scheduler

```
root@172 ~/oslab3> ./testbench.sh 2
Dummy run ---------
Loop0 ---------
14, 46, 179 io
13, 41, 158 io
14, 50, 166 io
17, 122, 0 cpu
17, 126, 0 cpu
17, 207, 0 cpu
Loop1 ---------
10, 37, 154 io
10, 34, 119 io
10, 34, 156 io
17, 176, 0 cpu
17, 163, 0 cpu
18, 186, 0 cpu
Testbench done!
root@172 ~/oslab3> _
```

new scheduler

```
root@172 ~/oslab3> ./testbench.sh 2
Dummy run ---------
Loop0 ---------
11, 140, 0 cpu
17, 32, 169 io
16, 45, 137 io
16, 38, 169 io
17, 136, 0 cpu
16, 134, 0 cpu
Loop1 ---------
10, 116, 0 cpu
10, 123, 0 cpu
17, 33, 180 io
17, 41, 193 io
16, 56, 178 io
17, 132, 0 cpu
Testbench done!
root@172 ~/oslab3> _
```

Σημειώσεις:
- Το γεγονός ότι δεν ολοκληρώνονται πάντα πρώτα οι cpu διεργασίες οφείλεται στο ότι το cpu_bound πρόγραμμα είναι εν γένει πιο αργό απο το io_bound.

- Το testbench που παρέχεται δεν εκτυπώνει τους χρόνους στην οθόνη αλλά για το κάθε πρόγραμμα ξεχωριστά γράφει τους χρόνους σε αρχείο μέσα στον φάκελο output.
Απο εκεί η επεξεργασία των τιμών real (πρώτη στήλη) δίνει τους παρακάτω πίνακες.

4. Πίνακες αποτελεσμάτων εκτελέσεων

cpu_bound.c

----- Default Algorithm ----			---- Modified Algorithm ----		
#	Inst1	Inst2	Inst3	Inst1	Inst2	Inst3
1	15	15	15	9	10	15
2	16	16	16	16	16	16
3	16	18	18	11	18	18
4	17	17	17	10	17	17
5	18	19	20	10	11	17
6	17	17	17	19	20	19
7	20	19	20	10	17	16
8	17	17	18	18	18	18
9	19	19	18	10	10	10
10	20	19	19	11	17	18
11	16	18	17	16	16	16
12	19	19	19	11	18	17
13	19	20	19	18	18	19
14	18	17	17	19	20	20
15	20	20	20	10	16	15
16	18	17	18	19	19	19
17	20	19	19	11	10	16
18	17	17	16	12	17	18
19	19	19	19	11	10	10
20	20	19	20	20	20	20
Av:	18.050	18.050	18.100	13.550	15.900	16.700
Dev:	1.564	1.322	1.446	3.879	3.506	2.666
Total Average:			18.067			15.383

io_bound.c

----- Default Algorithm ----			---- Modified Algorithm ----		
#	Inst1	Inst2	Inst3	Inst1	Inst2	Inst3
1	9	9	9	15	15	15
2	12	13	13	15	15	16
3	13	13	13	17	18	18
4	12	12	12	17	18	17
5	13	14	13	16	17	17
6	10	10	10	18	18	18
7	14	14	14	17	18	17
8	14	14	14	16	16	16
9	12	13	12	16	17	17
10	13	14	14	17	17	17
11	10	10	10	16	17	17
12	13	13	14	18	18	18
13	14	14	14	17	18	17
14	14	13	13	19	19	19
15	15	15	15	15	18	17
16	13	13	14	17	19	19
17	14	14	14	16	16	16
18	14	14	15	18	18	19
19	12	12	12	17	17	17
20	15	15	15	18	19	19
Av:	12.800	12.950	13.000	16.750	17.400	17.300
Dev:	1.600	1.596	1.673	1.090	1.158	1.100
Total Average:			12.917			17.150

5. Σχόλια για την συμπεριφορά του αλγορίθμου

Ο user-space scheduler παρέχει την ευκολία για επαναπροσδιορισμό της πολιτικής του χωρίς να επηρεάζεται ο kernel και το υπόλοιπο σύστημα, επομένως ο στόχος ήταν οι αλλαγές να είναι εξολοκλήρου σε αυτόν. Η συγκεκριμένη αλλαγή στον scheduler του λειτουργικού έχει σκοπό να μειώσει την προτεραιότητα των io_bound διεργασιών αφήνοντας την υπόλοιπη πολιτική του scheduler ανέπαφη.

Ο διαχωρισμός των cpu_bound και io_bound διεργασιών και το rescheduling των io_bound μπορεί να πραγματοποιηθεί σε πολλαπλά σημεία στον κώδικα του scheduler. Κάθε φορά που μια διεργασία μένει απο quantum , καλείται η do_noquantum η οποία κατεβάζει κατα ένα το priority της διεργασίας ώστε να μην μονοπολεί τον επεξεργαστή. Η balance_queues καλείται περιοδικά κ ανεβάζει την προτεραιότητα των διεργασιών που έχουν μειωθεί ώστε να μην μένουν χωρίς πόρους.

Η επιπρόσθετη λειτουργικότητα προστέθηκε στην do_noquantum. Όταν τελειώνει το quantum μιας διεργασίας αποφασίζεται σε αυτή την συνάρτηση αν κατανάλωσε όλο το quantum της (cpu_bound) ή εαν πραγματοποίησε blocking io calls κατα την ανάθεση της για εκτέλεση(io_bound). Ο διαχωρισμός αυτός γίνεται χρησιμοποιώντας τους χρόνους της εκάστοτε διεργασίας απο τον πίνακα proc του kernel μέσω της sys_getinfo(). Αν μια διεργασία χρησιμοποίησε πόρους συστήματος αναλογικά για περισσότερο χρόνο από ότι εκτελέστηκε στον kernel (stime/utime>1) τότε επιλέγεται ως io_bound διεργασία, μεταφέρεται στην τελευταία ουρά προτεραιότητας και γίνεται reschedule.

Αυτό έχει ώς αποτέλεσμα οι cpu να προτιμούνται έναντι των io bound διεργασιών όπως φαίνεται και στα αποτελέσματα. Ο μέσος όρος των cpu διεργασιών μειώνεται ενώ των io αυξάνει.