Λειτουργικά Συστήματα

2024 - 2025

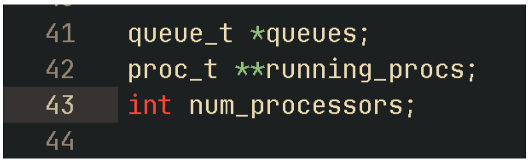
2η Εργαστηριακή Άσκηση (Project)

Κωνσταντίνος Αλεξίου, ΑΜ:1058083, up1058083@ac.upatras.gr
Παναγιώτα Γκιριτζιώνη, ΑΜ:1070953, up1070953@ac.upatras.gr
Αχιλλέας Πλιάτσικας, ΑΜ:1070946, up1070946@ac.upatras.gr
Γεώργιος Τσάμης, ΑΜ:1084659, up1084659@ac.upatras.gr

Πρώτη Φάση: υποστήριξη πολλαπλών επεξεργαστών

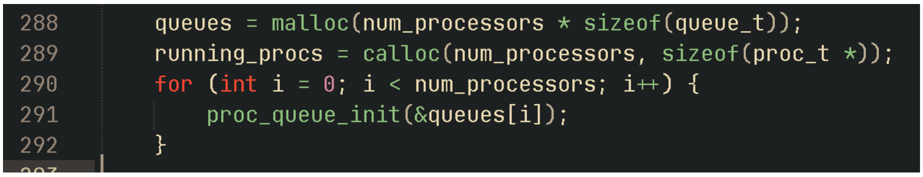
Εξήγηση προγράμματος

Το τροποποιημένο πρόγραμμα χρησιμοποιεί πολλαπλές ουρές (μία για κάθε επεξεργαστή) για την αποθήκευση των διεργασιών. Ο πίνακας queues διαχειρίζεται αυτές τις ουρές, ενώ ο πίνακας running_procs παρακολουθεί ποια διεργασία εκτελείται σε κάθε επεξεργαστή.
Προστέθηκαν οι εξής μεταβλητές:



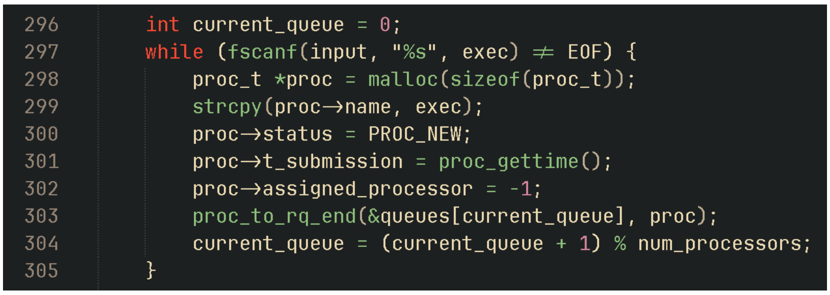
```c
queue_t *queues;
proc_t **running_procs;
int num_processors;
```

Κάνουμε αρχικοποίηση:



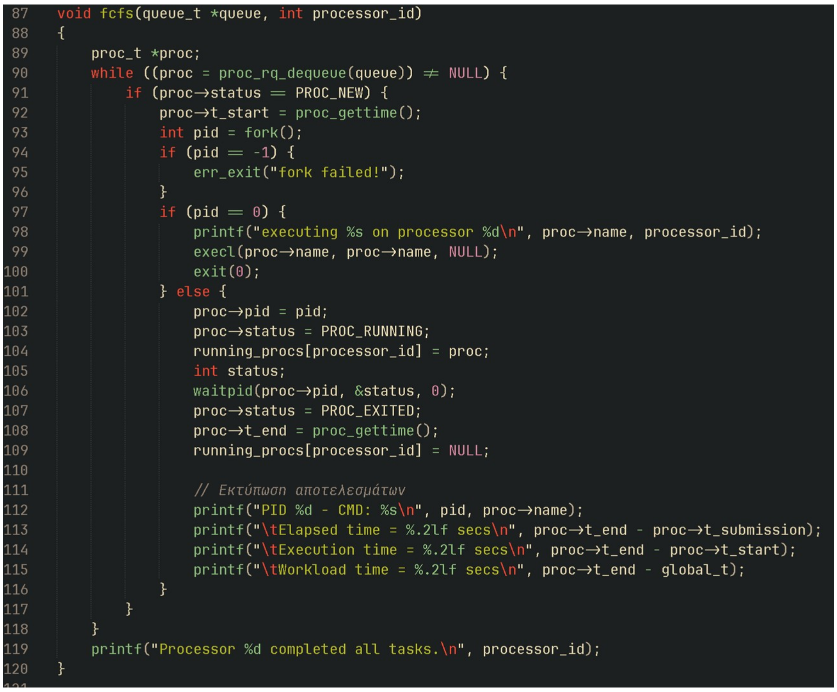
```c
    queues = malloc(num_processors * sizeof(queue_t));
    running_procs = calloc(num_processors, sizeof(proc_t *));
    for (int i = 0; i < num_processors; i++) {
        proc_queue_init(&queues[i]);
    }
```

Κατανομή διεργασιών στις ουρές:

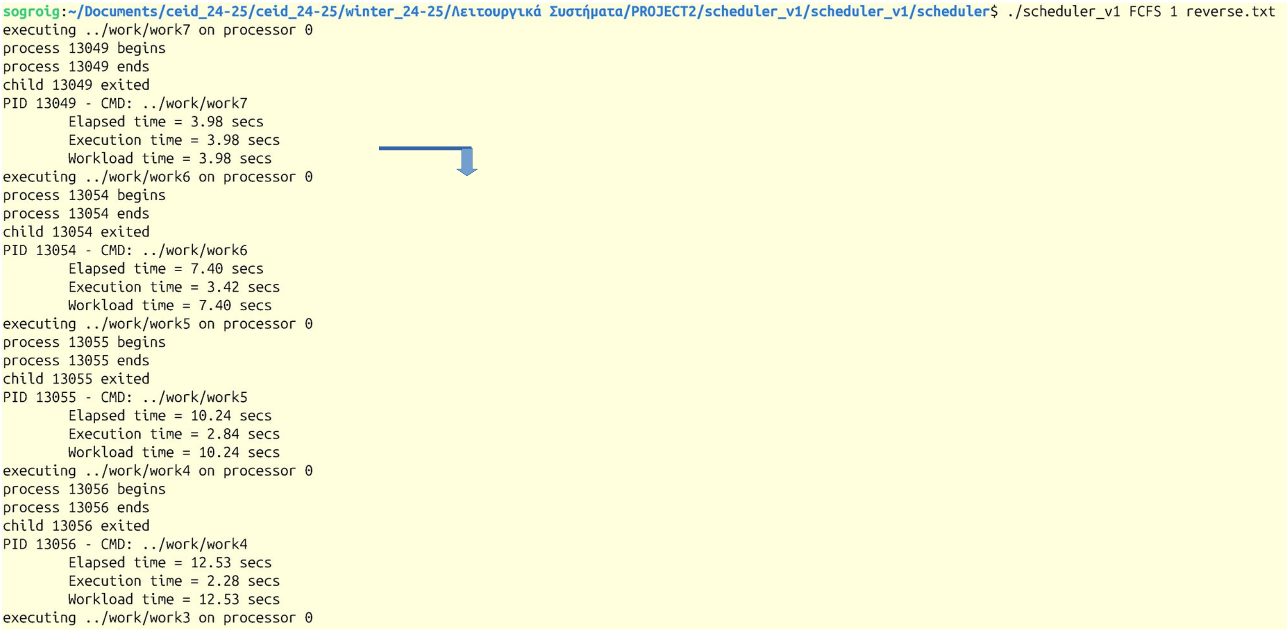


```c
    int current_queue = 0;
    while (fscanf(input, "%s", exec) != EOF) {
        proc_t *proc = malloc(sizeof(proc_t));
        strcpy(proc->name, exec);
        proc->status = PROC_NEW;
        proc->t_submission = proc_gettime();
        proc->assigned_processor = -1;
        proc_to_rq_end(&queues[current_queue], proc);
        current_queue = (current_queue + 1) % num_processors;
    }
```

Η πολιτική FCFS εκτελεί διεργασίες με τη σειρά που βρίσκονται στην ουρά, χωρίς χρονικό καταμερισμό. Ενημερώθηκε η συνάρτηση fcfs για υποστήριξη πολλαπλών επεξεργαστών:
Δοκιμάζουμε τον FCFS για εναν επεξργαστη



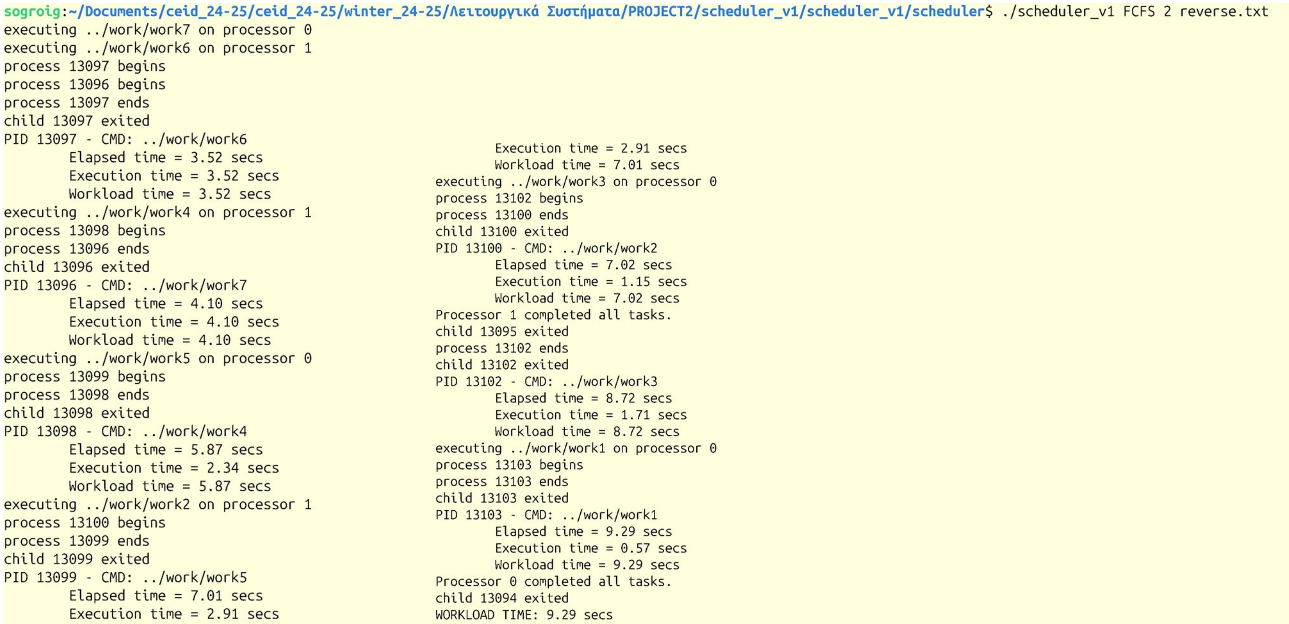
```c
void fcfs(queue_t *queue, int processor_id)
{
    proc_t *proc;
    while ((proc = proc_rq_dequeue(queue)) != NULL) {
        if (proc->status == PROC_NEW) {
            proc->t_start = proc_gettime();
            int pid = fork();
            if (pid == -1) {
                err_exit("fork failed!");
            }
            if (pid == 0) {
                printf("executing %s on processor %d\n", proc->name, processor_id);
                execl(proc->name, proc->name, NULL);
                exit(0);
            } else {
                proc->pid = pid;
                proc->status = PROC_RUNNING;
                running_procs[processor_id] = proc;
                int status;
                waitpid(proc->pid, &status, 0);
                proc->status = PROC_EXITED;
                proc->t_end = proc_gettime();
                running_procs[processor_id] = NULL;

                // Εκτύπωση αποτελεσμάτων
                printf("PID %d - CMD: %s\n", pid, proc->name);
                printf("\tElapsed time = %.2lf secs\n", proc->t_end - proc->t_submission);
                printf("\tExecution time = %.2lf secs\n", proc->t_end - proc->t_start);
                printf("\tWorkload time = %.2lf secs\n", proc->t_end - global_t);
            }
        }
    }
    printf("Processor %d completed all tasks.\n", processor_id);
}
```



```
sogroig:~/Documents/ceid_24-25/ceid_24-25/winter_24-25/Λειτουργικά Συστήματα/PROJECT2/scheduler_v1/scheduler_v1/scheduler$ ./scheduler_v1 FCFS 1 reverse.txt
executing ../work/work7 on processor 0
process 13049 begins
process 13049 ends
child 13049 exited
PID 13049 - CMD: ../work/work7
        Elapsed time = 3.98 secs
        Execution time = 3.98 secs
        Workload time = 3.98 secs
executing ../work/work6 on processor 0
process 13054 begins
process 13054 ends
child 13054 exited
PID 13054 - CMD: ../work/work6
        Elapsed time = 7.40 secs
        Execution time = 3.42 secs
        Workload time = 7.40 secs
executing ../work/work5 on processor 0
process 13055 begins
process 13055 ends
child 13055 exited
PID 13055 - CMD: ../work/work5
        Elapsed time = 10.24 secs
        Execution time = 2.84 secs
        Workload time = 10.24 secs
executing ../work/work4 on processor 0
process 13056 begins
process 13056 ends
child 13056 exited
PID 13056 - CMD: ../work/work4
        Elapsed time = 12.53 secs
        Execution time = 2.28 secs
        Workload time = 12.53 secs
executing ../work/work3 on processor 0
```

Είναι 2 ξεχωριστά screenshots τα τοποθετήσαμε έτσι για να φαίνονται καλύτερα

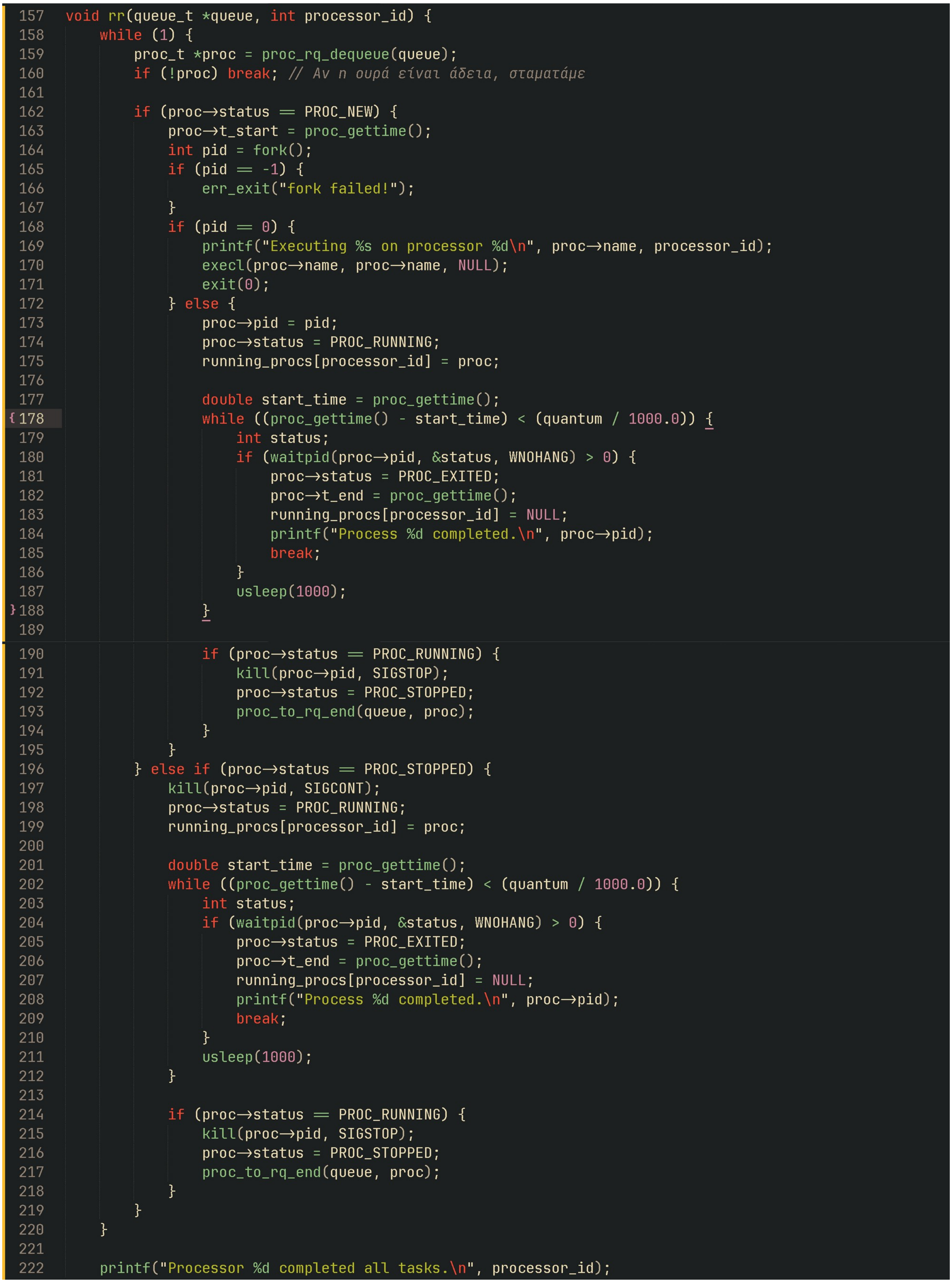
Και για 2 επεξεργαστές



```
sogroig:~/Documents/ceid_24-25/ceid_24-25/winter_24-25/Λειτουργικά Συστήματα/PROJECT2/scheduler_v1/scheduler_v1/scheduler$ ./scheduler_v1 FCFS 2 reverse.txt
executing ../work/work7 on processor 0
executing ../work/work6 on processor 1
process 13097 begins
process 13096 begins
process 13097 ends
child 13097 exited
PID 13097 - CMD: ../work/work6
        Elapsed time = 3.52 secs
        Execution time = 3.52 secs
        Workload time = 3.52 secs
executing ../work/work4 on processor 1
process 13098 begins
process 13096 ends
child 13096 exited
PID 13096 - CMD: ../work/work7
        Elapsed time = 4.10 secs
        Execution time = 4.10 secs
        Workload time = 4.10 secs
executing ../work/work5 on processor 0
process 13099 begins
process 13098 ends
child 13098 exited
PID 13098 - CMD: ../work/work4
        Elapsed time = 5.87 secs
        Execution time = 2.34 secs
        Workload time = 5.87 secs
executing ../work/work2 on processor 1
process 13100 begins
process 13099 ends
child 13099 exited
PID 13099 - CMD: ../work/work5
        Elapsed time = 7.01 secs
        Execution time = 2.91 secs
```

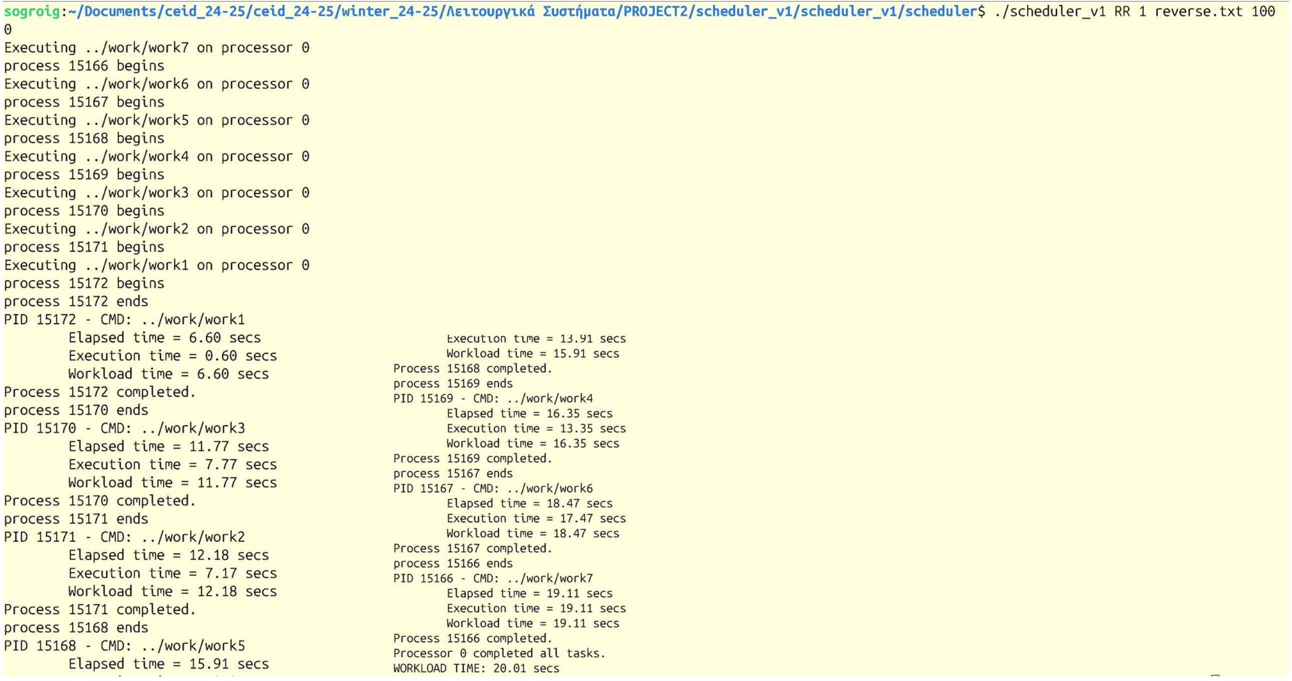
```
        Execution time = 2.91 secs
        Workload time = 7.01 secs
executing ../work/work3 on processor 0
process 13102 begins
process 13100 ends
child 13100 exited
PID 13100 - CMD: ../work/work2
        Elapsed time = 7.02 secs
        Execution time = 1.15 secs
        Workload time = 7.02 secs
Processor 1 completed all tasks.
child 13095 exited
process 13102 ends
child 13102 exited
PID 13102 - CMD: ../work/work3
        Elapsed time = 8.72 secs
        Execution time = 1.71 secs
        Workload time = 8.72 secs
executing ../work/work1 on processor 0
process 13103 begins
process 13103 ends
child 13103 exited
PID 13103 - CMD: ../work/work1
        Elapsed time = 9.29 secs
        Execution time = 0.57 secs
        Workload time = 9.29 secs
Processor 0 completed all tasks.
child 13094 exited
WORKLOAD TIME: 9.29 secs
```

Η πολιτική Round Robin εκτελεί διεργασίες για προκαθορισμένο χρονικό διάστημα (quantum). Αν μια διεργασία δεν ολοκληρωθεί, σταματά (SIGSTOP) και επιστρέφει στο τέλος της ουράς. Η συνάρτηση rr ενημερώθηκε για χρήση πολλαπλών επεξεργαστών:



```c
void rr(queue_t *queue, int processor_id) {
    while (1) {
        proc_t *proc = proc_rq_dequeue(queue);
        if (!proc) break; // Αν η ουρά είναι άδεια, σταματάμε

        if (proc->status == PROC_NEW) {
            proc->t_start = proc_gettime();
            int pid = fork();
            if (pid == -1) {
                err_exit("fork failed!");
            }
            if (pid == 0) {
                printf("Executing %s on processor %d\n", proc->name, processor_id);
                execl(proc->name, proc->name, NULL);
                exit(0);
            } else {
                proc->pid = pid;
                proc->status = PROC_RUNNING;
                running_procs[processor_id] = proc;

                double start_time = proc_gettime();
                while ((proc_gettime() - start_time) < (quantum / 1000.0)) {
                    int status;
                    if (waitpid(proc->pid, &status, WNOHANG) > 0) {
                        proc->status = PROC_EXITED;
                        proc->t_end = proc_gettime();
                        running_procs[processor_id] = NULL;
                        printf("Process %d completed.\n", proc->pid);
                        break;
                    }
                    usleep(1000);
                }

                if (proc->status == PROC_RUNNING) {
                    kill(proc->pid, SIGSTOP);
                    proc->status = PROC_STOPPED;
                    proc_to_rq_end(queue, proc);
                }
            }
        } else if (proc->status == PROC_STOPPED) {
            kill(proc->pid, SIGCONT);
            proc->status = PROC_RUNNING;
            running_procs[processor_id] = proc;

            double start_time = proc_gettime();
            while ((proc_gettime() - start_time) < (quantum / 1000.0)) {
                int status;
                if (waitpid(proc->pid, &status, WNOHANG) > 0) {
                    proc->status = PROC_EXITED;
                    proc->t_end = proc_gettime();
                    running_procs[processor_id] = NULL;
                    printf("Process %d completed.\n", proc->pid);
                    break;
                }
                usleep(1000);
            }

            if (proc->status == PROC_RUNNING) {
                kill(proc->pid, SIGSTOP);
                proc->status = PROC_STOPPED;
                proc_to_rq_end(queue, proc);
            }
        }
    }

    printf("Processor %d completed all tasks.\n", processor_id);
```

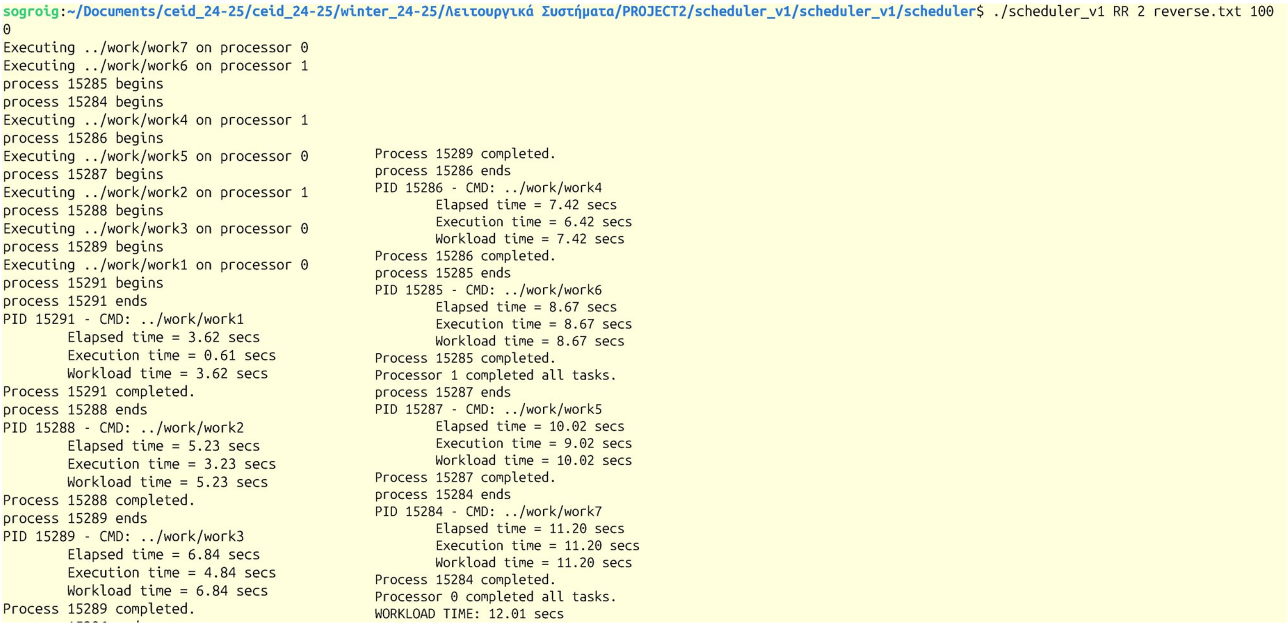
Δοκιμάζουμε τον RR για εναν επεξεργαστή



```
sogroig:~/Documents/ceid_24-25/ceid_24-25/winter_24-25/Λειτουργικά Συστήματα/PROJECT2/scheduler_v1/scheduler_v1/scheduler$ ./scheduler_v1 RR 1 reverse.txt 100
0
Executing ../work/work7 on processor 0
process 15166 begins
Executing ../work/work6 on processor 0
process 15167 begins
Executing ../work/work5 on processor 0
process 15168 begins
Executing ../work/work4 on processor 0
process 15169 begins
Executing ../work/work3 on processor 0
process 15170 begins
Executing ../work/work2 on processor 0
process 15171 begins
Executing ../work/work1 on processor 0
process 15172 begins
process 15172 ends
PID 15172 - CMD: ../work/work1
        Elapsed time = 6.60 secs
        Execution time = 0.60 secs
        Workload time = 6.60 secs
Process 15172 completed.
process 15170 ends
PID 15170 - CMD: ../work/work3
        Elapsed time = 11.77 secs
        Execution time = 7.77 secs
        Workload time = 11.77 secs
Process 15170 completed.
process 15171 ends
PID 15171 - CMD: ../work/work2
        Elapsed time = 12.18 secs
        Execution time = 7.17 secs
        Workload time = 12.18 secs
Process 15171 completed.
process 15168 ends
PID 15168 - CMD: ../work/work5
        Elapsed time = 15.91 secs
```

```
        Execution time = 13.91 secs
        Workload time = 15.91 secs
Process 15168 completed.
process 15169 ends
PID 15169 - CMD: ../work/work4
        Elapsed time = 16.35 secs
        Execution time = 13.35 secs
        Workload time = 16.35 secs
Process 15169 completed.
process 15167 ends
PID 15167 - CMD: ../work/work6
        Elapsed time = 18.47 secs
        Execution time = 17.47 secs
        Workload time = 18.47 secs
Process 15167 completed.
process 15166 ends
PID 15166 - CMD: ../work/work7
        Elapsed time = 19.11 secs
        Execution time = 19.11 secs
        Workload time = 19.11 secs
Process 15166 completed.
Processor 0 completed all tasks.
WORKLOAD TIME: 20.01 secs
```

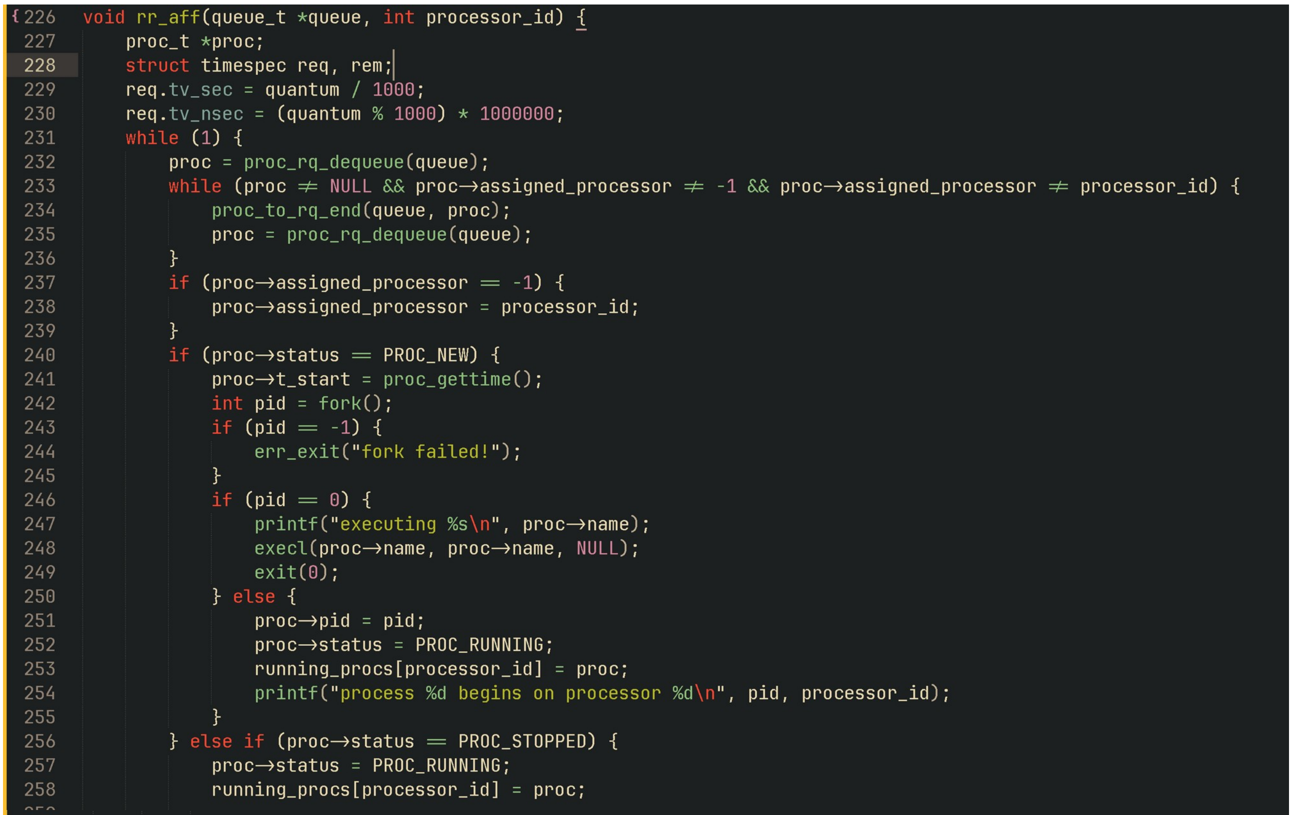
Και για 2 επεξεργαστές



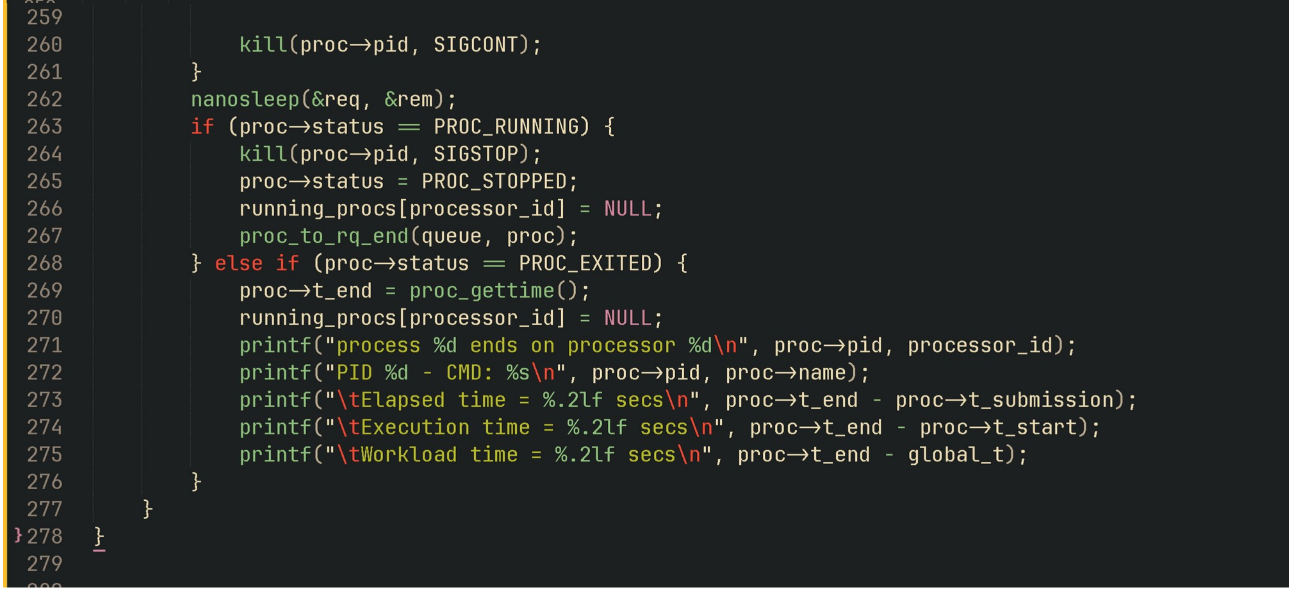
```
sogroig:~/Documents/ceid_24-25/ceid_24-25/winter_24-25/Λειτουργικά Συστήματα/PROJECT2/scheduler_v1/scheduler_v1/scheduler$ ./scheduler_v1 RR 2 reverse.txt 100
0
Executing ../work/work7 on processor 0
Executing ../work/work6 on processor 1
process 15285 begins
process 15284 begins
Executing ../work/work4 on processor 1
process 15286 begins
Executing ../work/work5 on processor 0
process 15287 begins
Executing ../work/work2 on processor 1
process 15288 begins
Executing ../work/work3 on processor 0
process 15289 begins
Executing ../work/work1 on processor 0
process 15291 begins
process 15291 ends
PID 15291 - CMD: ../work/work1
        Elapsed time = 3.62 secs
        Execution time = 0.61 secs
        Workload time = 3.62 secs
Process 15291 completed.
process 15288 ends
PID 15288 - CMD: ../work/work2
        Elapsed time = 5.23 secs
        Execution time = 3.23 secs
        Workload time = 5.23 secs
Process 15288 completed.
process 15289 ends
PID 15289 - CMD: ../work/work3
        Elapsed time = 6.84 secs
        Execution time = 4.84 secs
        Workload time = 6.84 secs
Process 15289 completed.
```

```
Process 15289 completed.
process 15286 ends
PID 15286 - CMD: ../work/work4
        Elapsed time = 7.42 secs
        Execution time = 6.42 secs
        Workload time = 7.42 secs
Process 15286 completed.
process 15285 ends
PID 15285 - CMD: ../work/work6
        Elapsed time = 8.67 secs
        Execution time = 8.67 secs
        Workload time = 8.67 secs
Process 15285 completed.
Processor 1 completed all tasks.
process 15287 ends
PID 15287 - CMD: ../work/work5
        Elapsed time = 10.02 secs
        Execution time = 9.02 secs
        Workload time = 10.02 secs
Process 15287 completed.
process 15284 ends
PID 15284 - CMD: ../work/work7
        Elapsed time = 11.20 secs
        Execution time = 11.20 secs
        Workload time = 11.20 secs
Process 15284 completed.
Processor 0 completed all tasks.
WORKLOAD TIME: 12.01 secs
```

Η πολιτική **RR-AFF** εκτελεί διεργασίες με affinity προς συγκεκριμένο επεξεργαστή. Μια διεργασία εκχωρείται σε επεξεργαστή κατά την πρώτη της ανάθεση και προστέθηκε η συνάρτηση rr_aff:

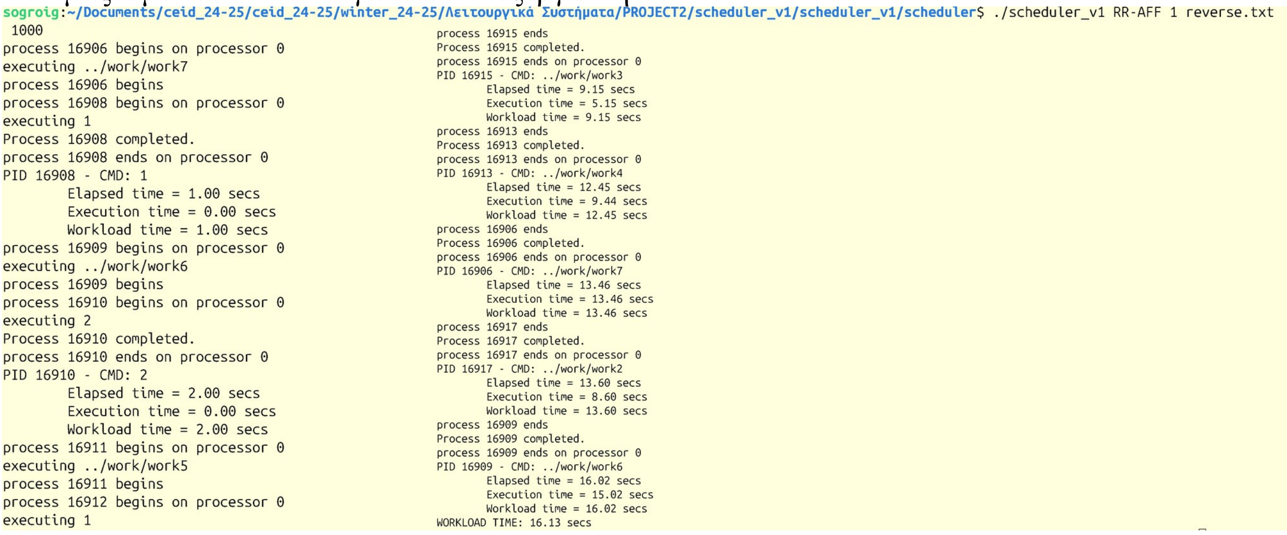


```c
void rr_aff(queue_t *queue, int processor_id) {
    proc_t *proc;
    struct timespec req, rem;
    req.tv_sec = quantum / 1000;
    req.tv_nsec = (quantum % 1000) * 1000000;
    while (1) {
        proc = proc_rq_dequeue(queue);
        while (proc != NULL && proc->assigned_processor != -1 && proc->assigned_processor != processor_id) {
            proc_to_rq_end(queue, proc);
            proc = proc_rq_dequeue(queue);
        }
        if (proc->assigned_processor == -1) {
            proc->assigned_processor = processor_id;
        }
        if (proc->status == PROC_NEW) {
            proc->t_start = proc_gettime();
            int pid = fork();
            if (pid == -1) {
                err_exit("fork failed!");
            }
            if (pid == 0) {
                printf("executing %s\n", proc->name);
                execl(proc->name, proc->name, NULL);
                exit(0);
            } else {
                proc->pid = pid;
                proc->status = PROC_RUNNING;
                running_procs[processor_id] = proc;
                printf("process %d begins on processor %d\n", pid, processor_id);
            }
        } else if (proc->status == PROC_STOPPED) {
            proc->status = PROC_RUNNING;
            running_procs[processor_id] = proc;
```



```c

            kill(proc->pid, SIGCONT);
        }
        nanosleep(&req, &rem);
        if (proc->status == PROC_RUNNING) {
            kill(proc->pid, SIGSTOP);
            proc->status = PROC_STOPPED;
            running_procs[processor_id] = NULL;
            proc_to_rq_end(queue, proc);
        } else if (proc->status == PROC_EXITED) {
            proc->t_end = proc_gettime();
            running_procs[processor_id] = NULL;
            printf("process %d ends on processor %d\n", proc->pid, processor_id);
            printf("PID %d - CMD: %s\n", proc->pid, proc->name);
            printf("\tElapsed time = %.2lf secs\n", proc->t_end - proc->t_submission);
            printf("\tExecution time = %.2lf secs\n", proc->t_end - proc->t_start);
            printf("\tWorkload time = %.2lf secs\n", proc->t_end - global_t);
        }
    }
}

```

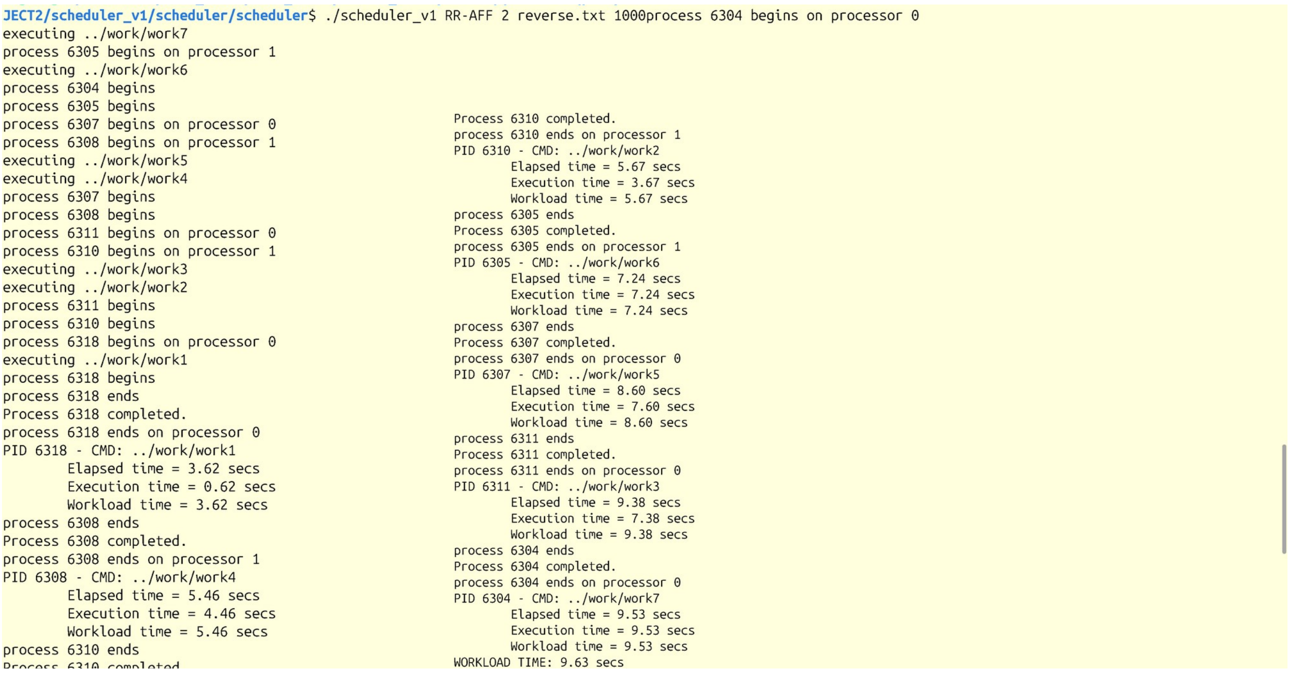
Δοκιμάζουμε τον RR-AFF για εναν επεξεργαστή



```
sogroig:~/Documents/ceid_24-25/ceid_24-25/winter_24-25/Λειτουργικά Συστήματα/PROJECT2/scheduler_v1/scheduler_v1/scheduler$ ./scheduler_v1 RR-AFF 1 reverse.txt
 1000
process 16906 begins on processor 0
executing ../work/work7
process 16906 begins
process 16908 begins on processor 0
executing 1
Process 16908 completed.
process 16908 ends on processor 0
PID 16908 - CMD: 1
        Elapsed time = 1.00 secs
        Execution time = 0.00 secs
        Workload time = 1.00 secs
process 16909 begins on processor 0
executing ../work/work6
process 16909 begins
process 16910 begins on processor 0
executing 2
Process 16910 completed.
process 16910 ends on processor 0
PID 16910 - CMD: 2
        Elapsed time = 2.00 secs
        Execution time = 0.00 secs
        Workload time = 2.00 secs
process 16911 begins on processor 0
executing ../work/work5
process 16911 begins
process 16912 begins on processor 0
executing 1
```

```
process 16915 ends
Process 16915 completed.
process 16915 ends on processor 0
PID 16915 - CMD: ../work/work3
        Elapsed time = 9.15 secs
        Execution time = 5.15 secs
        Workload time = 9.15 secs
process 16913 ends
Process 16913 completed.
process 16913 ends on processor 0
PID 16913 - CMD: ../work/work4
        Elapsed time = 12.45 secs
        Execution time = 9.44 secs
        Workload time = 12.45 secs
process 16906 ends
Process 16906 completed.
process 16906 ends on processor 0
PID 16906 - CMD: ../work/work7
        Elapsed time = 13.46 secs
        Execution time = 13.46 secs
        Workload time = 13.46 secs
process 16917 ends
Process 16917 completed.
process 16917 ends on processor 0
PID 16917 - CMD: ../work/work2
        Elapsed time = 13.60 secs
        Execution time = 8.60 secs
        Workload time = 13.60 secs
process 16909 ends
Process 16909 completed.
process 16909 ends on processor 0
PID 16909 - CMD: ../work/work6
        Elapsed time = 16.02 secs
        Execution time = 15.02 secs
        Workload time = 16.02 secs
WORKLOAD TIME: 16.13 secs
```

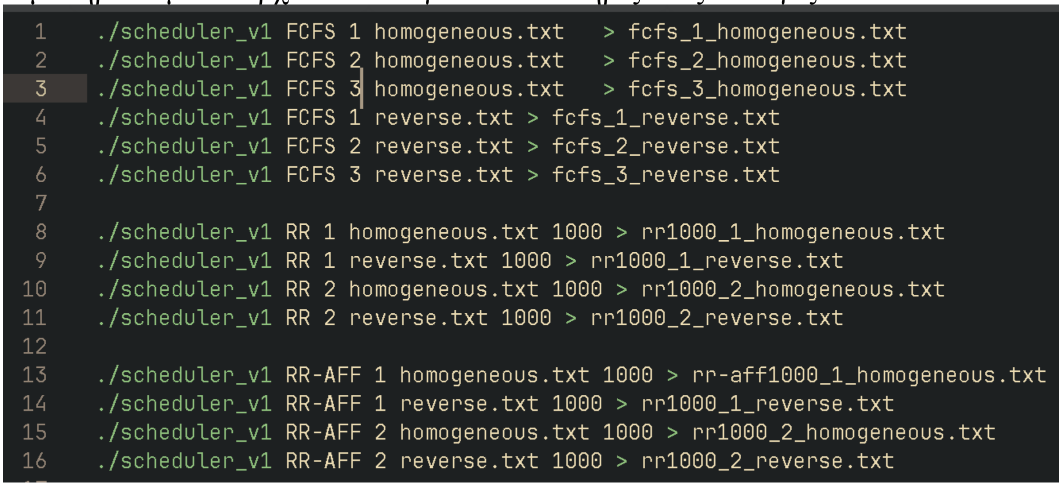
Και για 2 επεξεργαστές



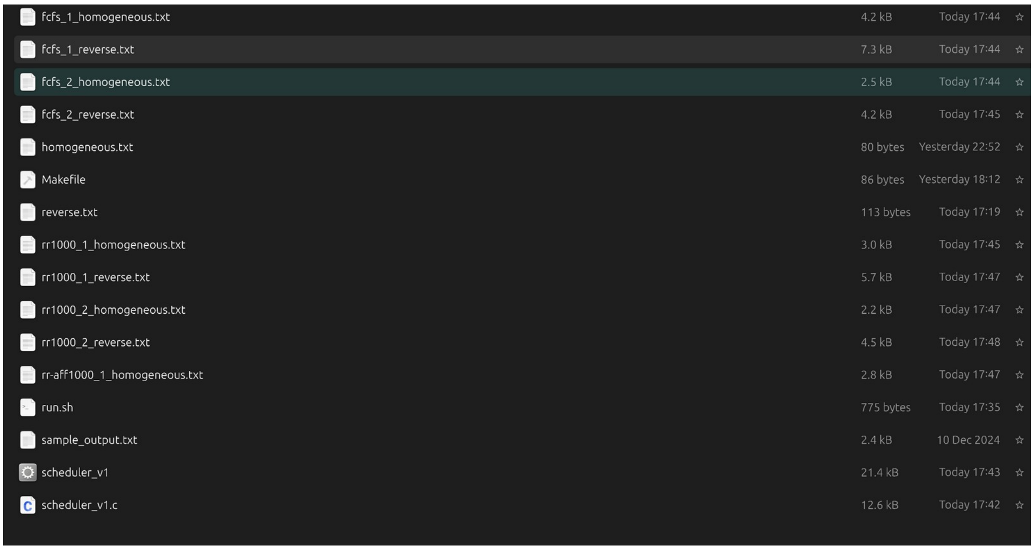
```
JECT2/scheduler_v1/scheduler/scheduler$ ./scheduler_v1 RR-AFF 2 reverse.txt 1000process 6304 begins on processor 0
executing ../work/work7
process 6305 begins on processor 1
executing ../work/work6
process 6304 begins
process 6305 begins
process 6307 begins on processor 0
process 6308 begins on processor 1
executing ../work/work5
executing ../work/work4
process 6307 begins
process 6308 begins
process 6311 begins on processor 0
process 6310 begins on processor 1
executing ../work/work3
executing ../work/work2
process 6311 begins
process 6310 begins
process 6318 begins on processor 0
executing ../work/work1
process 6318 begins
process 6318 ends
Process 6318 completed.
process 6318 ends on processor 0
PID 6318 - CMD: ../work/work1
        Elapsed time = 3.62 secs
        Execution time = 0.62 secs
        Workload time = 3.62 secs
process 6308 ends
Process 6308 completed.
process 6308 ends on processor 1
PID 6308 - CMD: ../work/work4
        Elapsed time = 5.46 secs
        Execution time = 4.46 secs
        Workload time = 5.46 secs
process 6310 ends
Process 6310 completed
```

```
Process 6310 completed.
process 6310 ends on processor 1
PID 6310 - CMD: ../work/work2
        Elapsed time = 5.67 secs
        Execution time = 3.67 secs
        Workload time = 5.67 secs
process 6305 ends
Process 6305 completed.
process 6305 ends on processor 1
PID 6305 - CMD: ../work/work6
        Elapsed time = 7.24 secs
        Execution time = 7.24 secs
        Workload time = 7.24 secs
process 6307 ends
Process 6307 completed.
process 6307 ends on processor 0
PID 6307 - CMD: ../work/work5
        Elapsed time = 8.60 secs
        Execution time = 7.60 secs
        Workload time = 8.60 secs
process 6311 ends
Process 6311 completed.
process 6311 ends on processor 0
PID 6311 - CMD: ../work/work3
        Elapsed time = 9.38 secs
        Execution time = 7.38 secs
        Workload time = 9.38 secs
process 6304 ends
Process 6304 completed.
process 6304 ends on processor 0
PID 6304 - CMD: ../work/work7
        Elapsed time = 9.53 secs
        Execution time = 9.53 secs
        Workload time = 9.53 secs
WORKLOAD TIME: 9.63 secs
```

Συμπληρώσαμε το αρχείο run.sh για να υποστηρίζει τις αλλαγές



```
./scheduler_v1 FCFS 1 homogeneous.txt   > fcfs_1_homogeneous.txt
./scheduler_v1 FCFS 2 homogeneous.txt   > fcfs_2_homogeneous.txt
./scheduler_v1 FCFS 3 homogeneous.txt   > fcfs_3_homogeneous.txt
./scheduler_v1 FCFS 1 reverse.txt > fcfs_1_reverse.txt
./scheduler_v1 FCFS 2 reverse.txt > fcfs_2_reverse.txt
./scheduler_v1 FCFS 3 reverse.txt > fcfs_3_reverse.txt

./scheduler_v1 RR 1 homogeneous.txt 1000 > rr1000_1_homogeneous.txt
./scheduler_v1 RR 1 reverse.txt 1000 > rr1000_1_reverse.txt
./scheduler_v1 RR 2 homogeneous.txt 1000 > rr1000_2_homogeneous.txt
./scheduler_v1 RR 2 reverse.txt 1000 > rr1000_2_reverse.txt

./scheduler_v1 RR-AFF 1 homogeneous.txt 1000 > rr-aff1000_1_homogeneous.txt
./scheduler_v1 RR-AFF 1 reverse.txt 1000 > rr1000_1_reverse.txt
./scheduler_v1 RR-AFF 2 homogeneous.txt 1000 > rr1000_2_homogeneous.txt
./scheduler_v1 RR-AFF 2 reverse.txt 1000 > rr1000_2_reverse.txt
```



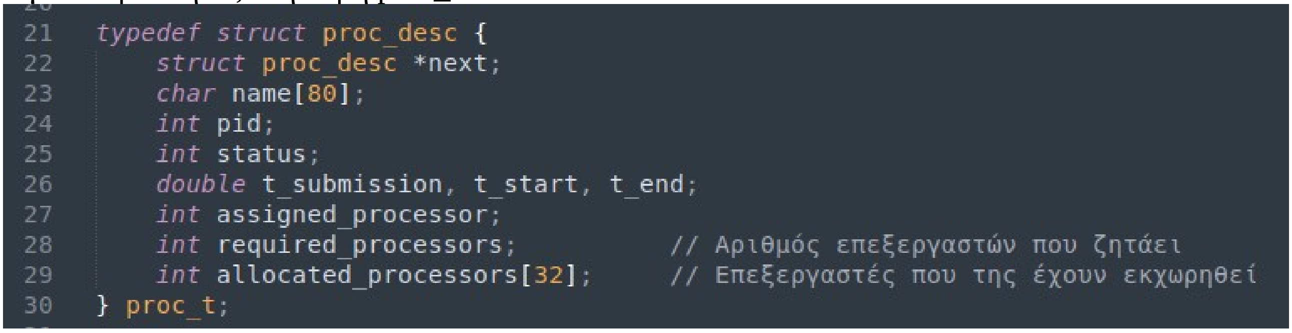
fcfs_1_homogeneous.txt	4.2 kB	Today 17:44
fcfs_1_reverse.txt	7.3 kB	Today 17:44
fcfs_2_homogeneous.txt	2.5 kB	Today 17:44
fcfs_2_reverse.txt	4.2 kB	Today 17:45
homogeneous.txt	80 bytes	Yesterday 22:52
Makefile	86 bytes	Yesterday 18:12
reverse.txt	113 bytes	Today 17:19
rr1000_1_homogeneous.txt	3.0 kB	Today 17:45
rr1000_1_reverse.txt	5.7 kB	Today 17:47
rr1000_2_homogeneous.txt	2.2 kB	Today 17:47
rr1000_2_reverse.txt	4.5 kB	Today 17:48
rr-aff1000_1_homogeneous.txt	2.8 kB	Today 17:47
run.sh	775 bytes	Today 17:35
sample_output.txt	2.4 kB	10 Dec 2024
scheduler_v1	21.4 kB	Today 17:43
scheduler_v1.c	12.6 kB	Today 17:42

Δεύτερη Φάση: Υποστήριξη αιτήσεων πολλαπλών επεξεργαστών

Εξήγηση Προγράμματος

Με βάση τον κώδικα που δημιουργήσαμε στην πρώτη φάση κάναμε κατάλληλες τροποποιήσεις και προσθήκες για την δεύτερη φάση, με σκοπό την υποστήριξη πολλαπλών επεξεργαστών για τον FCFS.

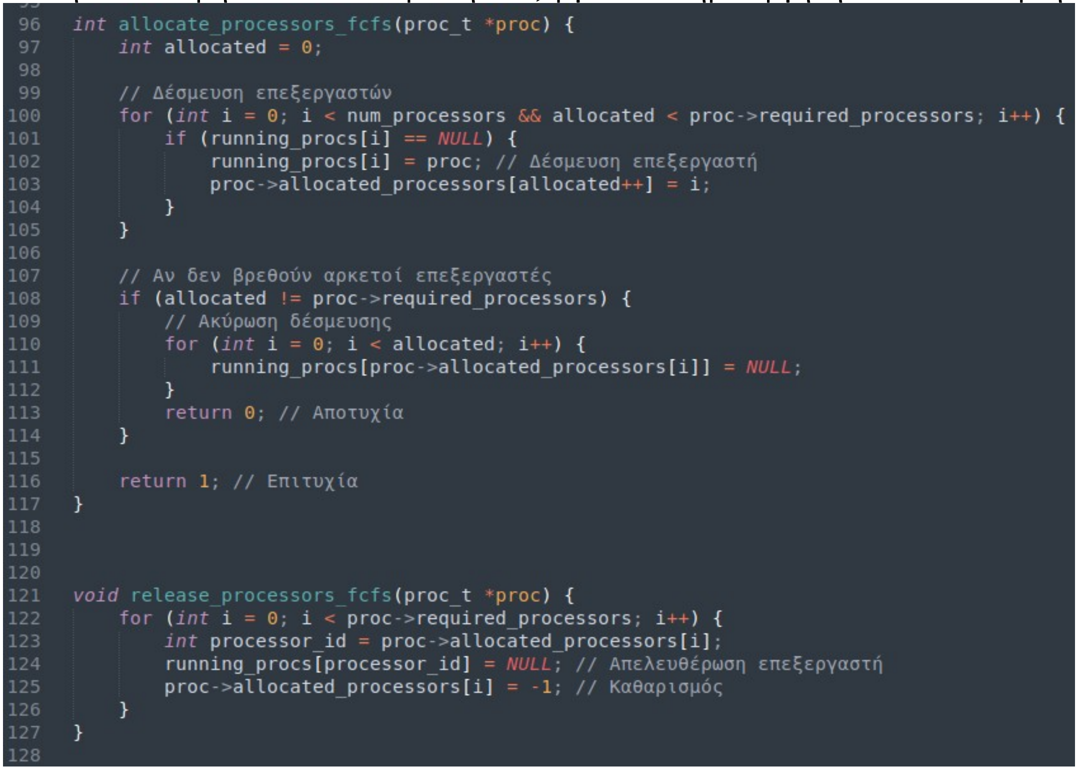
Έγιναν προσθήκες στη δομή proc_desc.



```c
typedef struct proc_desc {
    struct proc_desc *next;
    char name[80];
    int pid;
    int status;
    double t_submission, t_start, t_end;
    int assigned_processor;
    int required_processors;          // Αριθμός επεξεργαστών που ζητάει
    int allocated_processors[32];     // Επεξεργαστές που της έχουν εκχωρηθεί
} proc_t;
```

Προστέθηκαν οι ιδιότητες **required_processors & allocated_processors[32].**
Αυτές οι ιδιότητες προστέθηκαν για να υποστηρίξουν την εκχώρηση πολλαπλών επεξεργαστών σε μια διεργασία, ώστε να διαχειρίζονται καλύτερα οι απαιτήσεις πολλαπλών επεξεργαστών από κάθε διεργασία.

Για την κατανομή και απελευθέρωση επεξεργαστών δημιουργήθηκαν δύο συναρτήσεις.

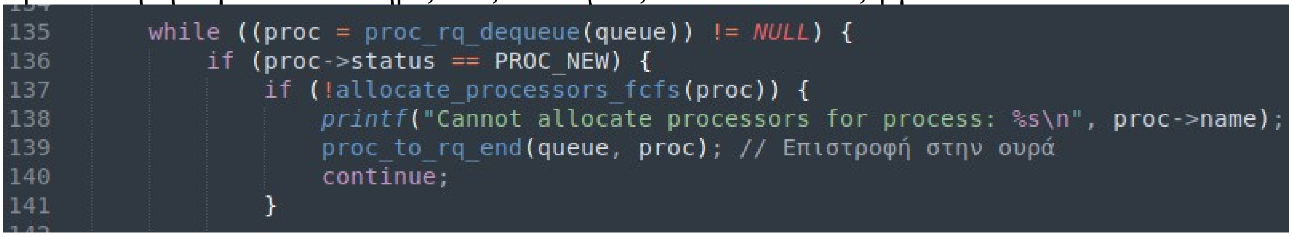


```c
int allocate_processors_fcfs(proc_t *proc) {
    int allocated = 0;

    // Δέσμευση επεξεργαστών
    for (int i = 0; i < num_processors && allocated < proc->required_processors; i++) {
        if (running_procs[i] == NULL) {
            running_procs[i] = proc; // Δέσμευση επεξεργαστή
            proc->allocated_processors[allocated++] = i;
        }
    }

    // Αν δεν βρεθούν αρκετοί επεξεργαστές
    if (allocated != proc->required_processors) {
        // Ακύρωση δέσμευσης
        for (int i = 0; i < allocated; i++) {
            running_procs[proc->allocated_processors[i]] = NULL;
        }
        return 0; // Αποτυχία
    }

    return 1; // Επιτυχία
}



void release_processors_fcfs(proc_t *proc) {
    for (int i = 0; i < proc->required_processors; i++) {
        int processor_id = proc->allocated_processors[i];
        running_procs[processor_id] = NULL; // Απελευθέρωση επεξεργαστή
        proc->allocated_processors[i] = -1; // Καθαρισμός
    }
}
```

allocate_processors_fcfs: Υλοποιεί τη δέσμευση των απαιτούμενων επεξεργαστών για κάθε διεργασία. Επιστρέφει 0 αν δεν υπάρχουν αρκετοί διαθέσιμοι επεξεργαστές, και αποδεσμεύει όσους είχαν δεσμευτεί μέχρι εκείνη τη στιγμή.

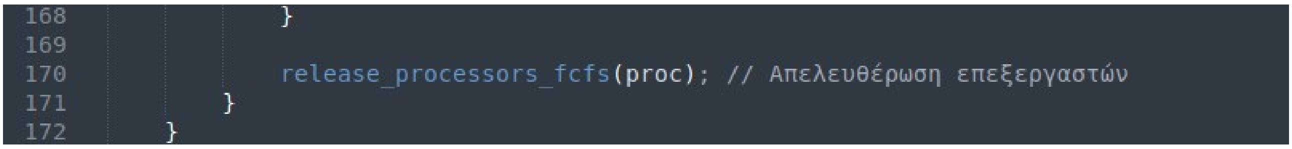
release_processors_fcfs: Απελευθερώνει τους επεξεργαστές που είχαν δεσμευτεί για μια διεργασία, ώστε να είναι διαθέσιμοι για άλλες διεργασίες.
Χρησιμοποιείται μετά την ολοκλήρωση της διεργασίας.

Ενημερώθηκε η **συνάρτηση FCFS.**
Τροποποιήθηκε για να υποστηρίζει τις απαιτήσεις πολλαπλών επεξεργαστών.



```
    while ((proc = proc_rq_dequeue(queue)) != NULL) {
        if (proc->status == PROC_NEW) {
            if (!allocate_processors_fcfs(proc)) {
                printf("Cannot allocate processors for process: %s\n", proc->name);
                proc_to_rq_end(queue, proc); // Επιστροφή στην ουρά
                continue;
            }
```

Προστέθηκε η χρήση της release_processors_fcfs μετά την ολοκλήρωση της διεργασίας.

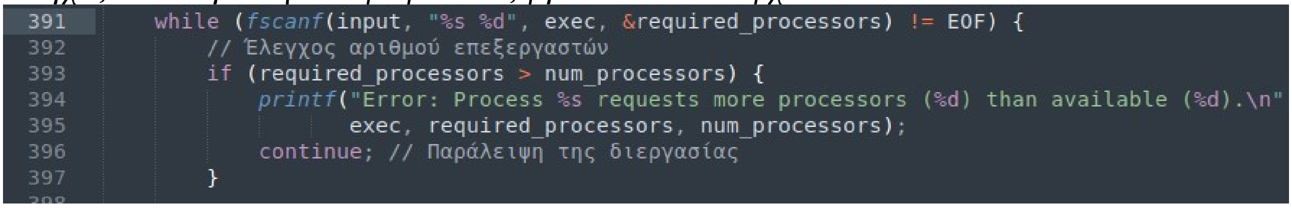


```
            }

            release_processors_fcfs(proc); // Απελευθέρωση επεξεργαστών
        }
    }
```

Η fcfs τώρα υποστηρίζει διεργασίες που απαιτούν περισσότερους από έναν επεξεργαστή. Αν δεν μπορεί να εκχωρηθεί ο απαιτούμενος αριθμός επεξεργαστών, η διεργασία επιστρέφεται στην ουρά.

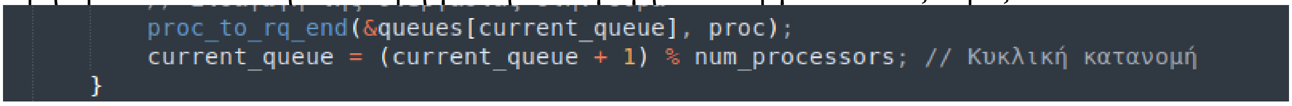
Ενημερώθηκε η **main,** ώστε να υποστηρίζει την ανάγνωση του αριθμού επεξεργαστών για κάθε διεργασία και την ανάθεση στις ουρές.

Έλεγχος και ανάγνωση του αριθμού επεξεργαστών από το αρχείο εισόδου.



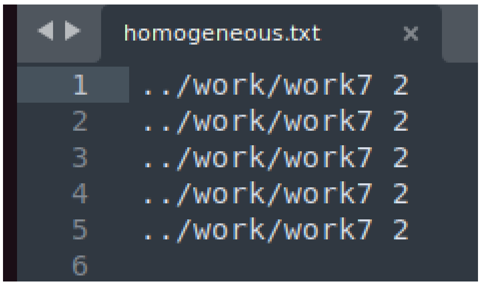
```
    while (fscanf(input, "%s %d", exec, &required_processors) != EOF) {
        // Έλεγχος αριθμού επεξεργαστών
        if (required_processors > num_processors) {
            printf("Error: Process %s requests more processors (%d) than available (%d).\n"
                    exec, required_processors, num_processors);
            continue; // Παράλειψη της διεργασίας
        }
```

Χρησιμοποιεί κυκλική κατανομή για την εισαγωγή των διεργασιών στις ουρές.

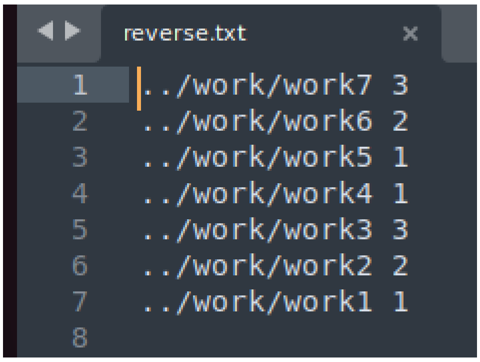


```
        proc_to_rq_end(&queues[current_queue], proc);
        current_queue = (current_queue + 1) % num_processors; // Κυκλική κατανομή
    }
```

Πρέπει να σημειωθεί ότι τα αρχεία εισόδου διαμορφώθηκαν ως εξής:

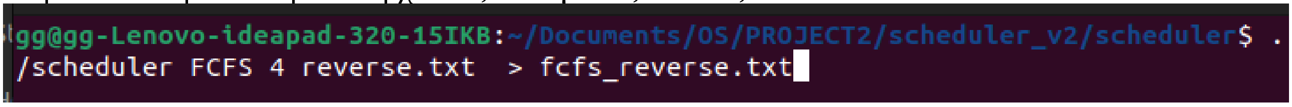


```
../work/work7 2
../work/work7 2
../work/work7 2
../work/work7 2
../work/work7 2

```



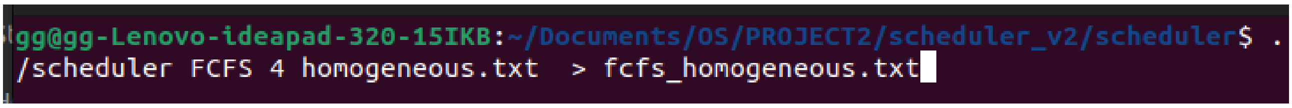
```
../work/work7 3
../work/work6 2
../work/work5 1
../work/work4 1
../work/work3 3
../work/work2 2
../work/work1 1

```

Κάθε διεργασία θέτει των απαιτούμενο αριθμό επεξεργαστών για τον εαυτό της.
Παρακάτω παραθέτουμε τα αρχεία εξόδου για τις εντολές :

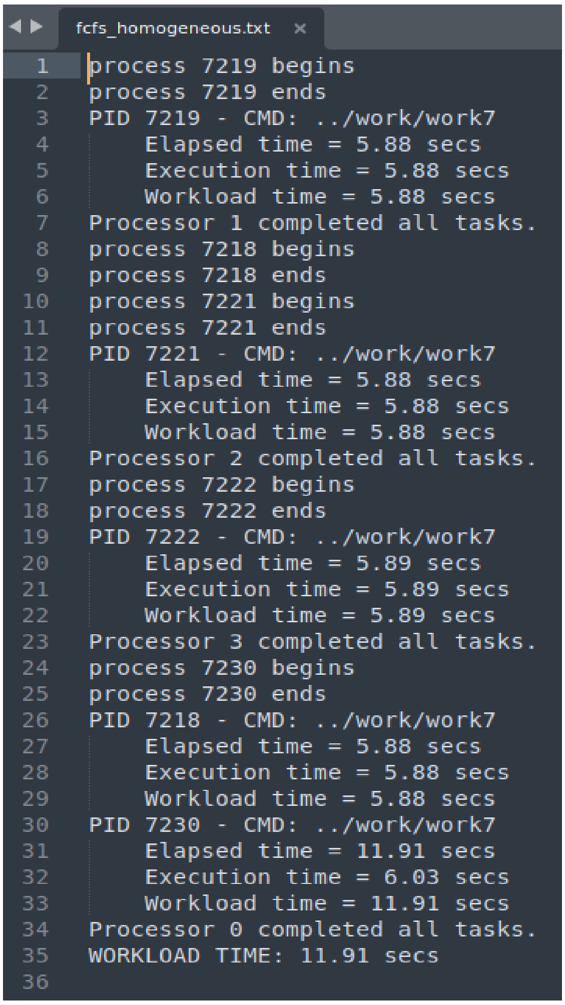


```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/OS/PROJECT2/scheduler_v2/scheduler$ .
/scheduler FCFS 4 reverse.txt  > fcfs_reverse.txt
```

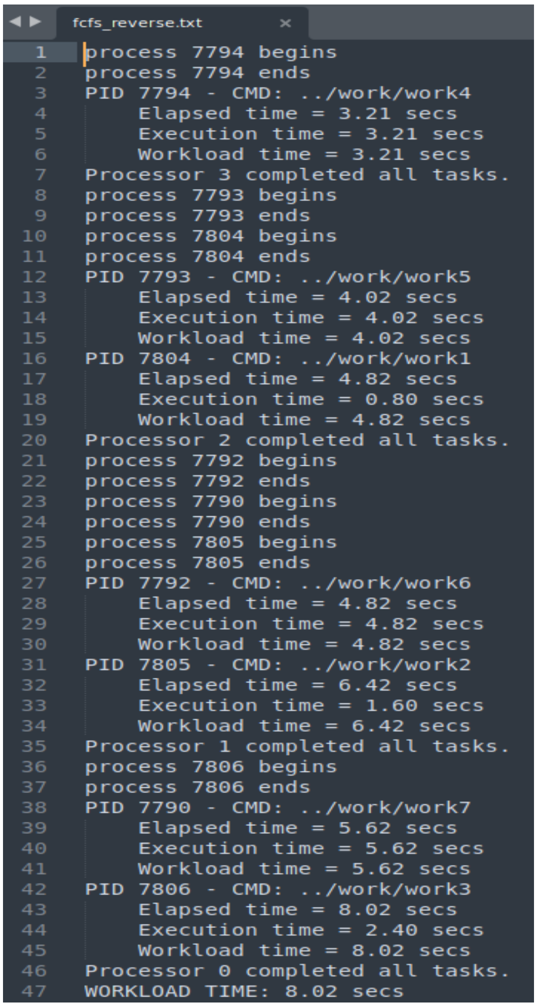
και



```
gg@gg-Lenovo-ideapad-320-15IKB:~/Documents/OS/PROJECT2/scheduler_v2/scheduler$ .
/scheduler FCFS 4 homogeneous.txt  > fcfs_homogeneous.txt
```



```
process 7219 begins
process 7219 ends
PID 7219 - CMD: ../work/work7
    Elapsed time = 5.88 secs
    Execution time = 5.88 secs
    Workload time = 5.88 secs
Processor 1 completed all tasks.
process 7218 begins
process 7218 ends
process 7221 begins
process 7221 ends
PID 7221 - CMD: ../work/work7
    Elapsed time = 5.88 secs
    Execution time = 5.88 secs
    Workload time = 5.88 secs
Processor 2 completed all tasks.
process 7222 begins
process 7222 ends
PID 7222 - CMD: ../work/work7
    Elapsed time = 5.89 secs
    Execution time = 5.89 secs
    Workload time = 5.89 secs
Processor 3 completed all tasks.
process 7230 begins
process 7230 ends
PID 7218 - CMD: ../work/work7
    Elapsed time = 5.88 secs
    Execution time = 5.88 secs
    Workload time = 5.88 secs
PID 7230 - CMD: ../work/work7
    Elapsed time = 11.91 secs
    Execution time = 6.03 secs
    Workload time = 11.91 secs
Processor 0 completed all tasks.
WORKLOAD TIME: 11.91 secs

```



```
process 7794 begins
process 7794 ends
PID 7794 - CMD: ../work/work4
    Elapsed time = 3.21 secs
    Execution time = 3.21 secs
    Workload time = 3.21 secs
Processor 3 completed all tasks.
process 7793 begins
process 7793 ends
process 7804 begins
process 7804 ends
PID 7793 - CMD: ../work/work5
    Elapsed time = 4.02 secs
    Execution time = 4.02 secs
    Workload time = 4.02 secs
PID 7804 - CMD: ../work/work1
    Elapsed time = 4.82 secs
    Execution time = 0.80 secs
    Workload time = 4.82 secs
Processor 2 completed all tasks.
process 7792 begins
process 7792 ends
process 7790 begins
process 7790 ends
process 7805 begins
process 7805 ends
PID 7792 - CMD: ../work/work6
    Elapsed time = 4.82 secs
    Execution time = 4.82 secs
    Workload time = 4.82 secs
PID 7805 - CMD: ../work/work2
    Elapsed time = 6.42 secs
    Execution time = 1.60 secs
    Workload time = 6.42 secs
Processor 1 completed all tasks.
process 7806 begins
process 7806 ends
PID 7790 - CMD: ../work/work7
    Elapsed time = 5.62 secs
    Execution time = 5.62 secs
    Workload time = 5.62 secs
PID 7806 - CMD: ../work/work3
    Elapsed time = 8.02 secs
    Execution time = 2.40 secs
    Workload time = 8.02 secs
Processor 0 completed all tasks.
WORKLOAD TIME: 8.02 secs
```