

# OPERATING SYSTEMS PROJECT 1

**-Κατσαρός Ανδρέας (1084522)**  
**-Ποταμιάνος Άγγελος Νικόλαος (1084537)**  
**-Τζωρτζάκης Γρηγόρης (1084538)**

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ: 2022-2023

—

ΕΡΓΑΣΤΗΡΙΟ ΛΕΙΤΟΥΡΓΙΚΩΝ  
ΣΥΣΤΗΜΑΤΩΝ

—

ΟΝΟΜΑΤΑ ΚΑΘΗΓΗΤΩΝ:  
Κος ΣΠΥΡΙΔΩΝ ΣΙΟΥΤΑΣ  
Κος ΧΡΗΣΤΟΣ ΜΑΚΡΗΣ  
Κος ΠΑΝΑΓΙΩΤΗΣ ΧΑΤΖΗΔΟΥΚΑΣ  
Κος ΑΡΙΣΤΕΙΔΗΣ ΗΛΙΑΣ

---

## **ΕΡΩΤΗΜΑΤΑ:**

### **ΕΡΩΤΗΜΑ 1: SHELL SCRIPTING**

#### **ΓΕΝΙΚΑ:**

Για την υλοποίηση του συγκεκριμένου ερωτήματος καθώς και του υπόλοιπου project (άσκηση 2) χρησιμοποιήσαμε το «Oracle VM Box» με την kali-linux-2022.3-virtualbox-amd64 διανομή.

Η προϋπόθεση το αρχείο να έχει “.log” προέκταση έγινε σε συνδυασμό με το πρώτο ερώτημα καθώς και με την υλοποίηση του case κορμού. Για να ελέγξουμε αν το αρχείο έχει την συγκεκριμένη προέκταση δεν χρειάζεται κάτι παραπάνω από το να ελέγξουμε αν το filename του είναι της μορφής \*.log (δηλαδή «οτιδήποτε που να καταλήγει σε

‘.log’»). Εφόσον το αρχείο μας είναι συμβατό και μας έχει δοθεί και τρίτο όρισμα, το script μας προχωράει στην ανάγνωση των case αλλιώς εμφανίζει στον χρήστη το μήνυμα: «Wrong File Argument»:

```
69
70 if [[ 0 = $# ]]; then
71     id
72
73 elif [[ $filename = *.log ]]; then
74     opt="$2";
75     case "$opt" in
76         "--usrid" )
77             mining_usernames "$3" ;;
78         "--method" ) method
79             "$3" ;;
80         "--browsers" )
81             count_browsers ;;
82         "--servprot" ) getip
83             "$3" ;;
84         "--datum" ) getdate
85             "$3" ;;
86         "" ) cat -n $1 ;;
87         * ) echo "Wrong
88             argument \o/ " ;;
89     esac
90 elif [[ $filename != *.log ]]; then
91     echo "Wrong File Argument"
92 fi
93
```

Η χρήση του \$filename (πρώτου ορίσματος στην τερματική κονσόλα) θα εξηγηθεί σε παρακάτω βήμα, όταν δηλαδή γίνεται και αναγκαία η δημιουργία του. Στην case δίνουμε στην μεταβλητή opt την τιμή του δεύτερου ορίσματος που θα εισάγουμε προκειμένου να μπορέσουμε να επιλέξουμε σε ποια κατηγορία της case βρισκόμαστε.

Εικόνα 1

Δοκιμάζοντας τον κώδικα παίρνουμε τα εξής αποτελέσματα:

```
(kali㉿kali)-[~/test]
$ ./logparser.sh test.jpeg
Wrong File Argument

(kali㉿kali)-[~/test]
$ ./logparser.sh log
Wrong File Argument

(kali㉿kali)-[~/test]
$ ./logparser.sh access
Wrong File Argument

(kali㉿kali)-[~/test]
$ ./logparser.sh access.
Wrong File Argument
```

(α)→Λάθος file extension

(β,γ)→Δεν δίνουμε file extension

(γ)→δίνουμε κενό file extension

Εικόνα 2

(i) Όπως φαίνεται στον κώδικα και από τα ζητούμενα της εκφώνησης, σε περίπτωση που δεν δοθούν ορίσματα στον τερματικό (δηλαδή αν \$#==0), τότε το πρόγραμμα μας θα καλέσει την συνάρτηση `id` η οποία περιέχει τα AM των μελών της ομάδας και έχει την συγκεκριμένη δομή.

```
12 id(){
13     echo "1084537|1084538|1084522"
14 }
```

Εικόνα 3

Δοκιμάζουμε να τρέξουμε το πρόγραμμα χωρίς ορίσματα και λαμβάνουμε το εξής επιθυμητό αποτέλεσμα:

```
(kali@kali)-[~/test]
$ ./logparser.sh
1084537|1084538|1084522
```

→ Εκτυπώνονται τα AM των μελών της ομάδας

Εικόνα 4

(ii) Στο δεύτερο υποερώτημα μας ζητείται όταν δίνεται ως πρώτο όρισμα το `filename` και κενό δεύτερο όρισμα (δηλαδή δίνεται σκέτο το όνομα του `.log` αρχείου), το script μας να εμφανίζει όλο το αρχείο ανά γραμμή. Αυτό επιτυγχάνεται με την χρήση της εντολής `cat` με όρισμα `-n` (numbered, δηλαδή αριθμημένες γραμμές) πάνω στο πρώτο όρισμα της εντολής που δώσαμε (δηλαδή το όνομα του αρχείου). Αυτό υλοποιείται στην γραμμή 81 της Εικόνας 1, παίρνοντας ως δεύτερο όρισμα από την case το κενό «" »». Σε αυτό το σημείο να αναφέρουμε πως στην συγκεκριμένη περίπτωση το `$1` (πρώτο όρισμα στην εντολή που γράφουμε στην κονσόλα) θα είχε το ίδιο αποτέλεσμα με το να γράφαμε την `$filename` μεταβλητή, απλά αυτό το τμήμα του κώδικα υλοποιήθηκε πριν την ανάγκη δημιουργίας της δεύτερης (που θα εξηγήσουμε στο επόμενο ερώτημα).



Έτσι με την εντολή “./logparser.sh access.log” λαμβάνουμε το εξής αποτέλεσμα:

```
(kali@kali)-[~/test]
$ ./logparser.sh access.log
1 127.0.0.1 - root - [29/Mar/2021:16:47:19 +0300] "GET / HTTP/1.1" 302 - "-" "Mozilla/5.0 (Windows NT 10.0; Wi
2 127.0.0.1 - - [29/Mar/2021:16:47:19 +0300] "GET /dashboard/ HTTP/1.1" 200 7577 "-" "Mozilla/5.0 (Windows NT
3 127.0.0.1 - admin - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/stylesheets/normalize.css HTTP/1.1" 200 687
4 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/stylesheets/all.css HTTP/1.1" 200 481698 "http://
5 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/javascripts/modernizr.js HTTP/1.1" 200 51365 "htt
6 127.0.0.1 - user1 - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/javascripts/all.js HTTP/1.1" 200 188385 "ht
7 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/bitnami-xampp.png HTTP/1.1" 200 22133 "htt
8 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/fastly-logo.png HTTP/1.1" 200 1770 "http:/
9 127.0.0.1 - root - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/xampp-logo.svg HTTP/1.1" 200 5427 "ht
10 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/social-icons.png HTTP/1.1" 200 3361 "http:
11 127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/favicon.png HTTP/1.1" 200 2508 "http://loc
12 127.0.0.1 - admin - [29/Mar/2021:16:47:21 +0300] "GET /phpmyadmin/ HTTP/1.1" 200 15865 "http://localhost/das
13 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/codemirror/addon/hint/show-hint.css?v=
14 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/codemirror/lib/codemirror.css?v=5.1.1
15 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/themes/pmahomme/jquery/jquery-ui.css HTTP/1.1" 2
16 127.0.0.1 - user1 - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/themes/pmahomme/css/theme.css?v=5.1.1&noca
17 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/sprintf.js?v=5.1.1 HTTP/1.1" 200 7409
18 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/codemirror/addon/lint/lint.css?v=5.1.1
19 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery-migrate.js?v=5.1.1 HTTP/
20 127.0.0.1 - root - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/dist/ajax.js?v=5.1.1 HTTP/1.1" 200 31313
21 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/dist/keyhandler.js?v=5.1.1 HTTP/1.1" 200 3003
22 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery.mousewheel.js?v=5.1.1 HT
23 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/js.cookie.js?v=5.1.1 HTTP/1.1" 200 388
24 127.0.0.1 - admin - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery.validate.js?v=5.1.
25 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery.min.js?v=5.1.1 HTTP/1.1
26 127.0.0.1 - root - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery.ba-hashchange-2.0.j
27 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery.debounce-1.0.6.js?v=5.1.
28 127.0.0.1 - user1 - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/dist/cross_framing_protection.js?v=5.1.
29 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/bootstrap/bootstrap.bundle.min.js?v=5.
30 127.0.0.1 - user2 - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/dist/menu_resizer.js?v=5.1.1 HTTP/1.1"
31 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/jquery/jquery-ui-timepicker-addon.js?v
32 127.0.0.1 - - [29/Mar/2021:16:47:24 +0300] "GET /phpmyadmin/js/vendor/tracekit.js?v=5.1.1 HTTP/1.1" 200 4538
```

Εικόνα 5: Αρχή του αρχείου

```
16266 ::1 - - [13/Oct/2022:04:47:08 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16267 ::1 - - [13/Oct/2022:11:34:12 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16268 ::1 - - [13/Oct/2022:11:52:25 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16269 ::1 - - [13/Oct/2022:12:05:22 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16270 ::1 - - [13/Oct/2022:12:18:06 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16271 ::1 - - [13/Oct/2022:12:30:38 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16272 ::1 - - [13/Oct/2022:12:42:57 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16273 ::1 - - [13/Oct/2022:12:55:04 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16274 ::1 - - [13/Oct/2022:13:06:59 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16275 ::1 - - [13/Oct/2022:13:42:16 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16276 ::1 - - [13/Oct/2022:13:53:49 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16277 ::1 - - [13/Oct/2022:14:05:12 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16278 ::1 - - [13/Oct/2022:14:16:24 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16279 ::1 - - [13/Oct/2022:14:27:24 +0300] "POST /phpmyadmin/index.php?route=/ HTTP/1.1" 200 1659
16280 ::1 - - [15/Oct/2022:19:27:55 +0300] "GET /MyDocs/ HTTP/1.1" 200 3948
16281 ::1 - - [15/Oct/2022:19:27:55 +0300] "GET /icons/blank.gif HTTP/1.1" 200 148
16282 ::1 - - [15/Oct/2022:19:27:55 +0300] "GET /icons/back.gif HTTP/1.1" 200 216
16283 ::1 - - [15/Oct/2022:19:27:55 +0300] "GET /icons/folder.gif HTTP/1.1" 200 225
16284 ::1 - - [15/Oct/2022:19:27:57 +0300] "GET /MyDocs/Vathmoi/ HTTP/1.1" 200 517
16285 ::1 - - [15/Oct/2022:19:27:59 +0300] "GET /MyDocs/ HTTP/1.1" 200 3948
16286 ::1 - - [15/Oct/2022:19:28:04 +0300] "GET /MyDocs/Software HTTP/1.1" 301 350
16287 ::1 - - [15/Oct/2022:19:28:04 +0300] "GET /MyDocs/Software/ HTTP/1.1" 200 519
16288 ::1 - - [15/Oct/2022:19:28:08 +0300] "GET /MyDocs/Software/after.php HTTP/1.1" 200 393
16289 ::1 - - [15/Oct/2022:19:28:10 +0300] "GET /MyDocs/Software/ HTTP/1.1" 200 519

(kali@kali)-[~/test]
$
```

Εικόνα 6: Τέλος του αρχείου access.log

(iii) Για αυτό το υποερώτημα, αρχικά από την case λαμβάνεται ως δεύτερο όρισμα \$2 η εντολή—usrid η οποία καλεί την συνάρτηση mining\_usernames() και εισάγει σε αυτή το τρίτο όρισμα (\$3) του υποερωτήματος (δηλαδή αν θέλουμε να επιλέξουμε ανάμεσα από admin, president, root, user1, user2, user3), στο οποίο μας εμφανίζει όσες

γραμμές υπάρχει μόνο ο συγκεκριμένος χρήστης. Η υλοποίηση της στην case φαίνεται στην γραμμή 76 της Εικόνας 1.

Η συνάρτηση `mining_usernames()` υλοποιείται ως εξής:

```
12 mining_usernames(){
13
14 var="$@"
15 if [[ -z $var ]]; then awk ' { arr[$3]++ } END
    { for( no in arr) { printf no "\t" arr[no] "\n"} } '
    $filename |sort | uniq
16
17 else awk -v pat="- $var" '$0~pat' $filename
18
19 fi
20 }
```

Εικόνα 7: `mining_usernames()`

Σε αυτό το σημείο, δεν μπορούσα να περάσω το πρώτο όρισμα (\$1) το οποίο είναι το όνομα του αρχείου που θέλω να προσπελάσω μέσα στην συνάρτηση, αφού η χρήση του \$1 μέσα στην συνάρτηση αναφέρεται στο πρώτο όρισμα που δίνεται στην παρένθεση της συνάρτησης. Για να λύσω το συγκεκριμένο πρόβλημα, πήγα στην κορυφή του script μου και όρισα ως global variable μια μεταβλητή με όνομα filename που παίρνει την τιμή του πρώτου ορίσματος με τον εξής τρόπο:

```
6 declare -g filename=$1
```

Έτσι όποτε χρησιμοποιώ την μεταβλητή filename αυτή θα παίρνει την τιμή του αρχείου μου, κάτι αρκετά χρήσιμο και για τα επόμενα υποερωτήματα και θα μπορώ να την αξιοποιώ μέσα στις συναρτήσεις, διότι είναι ορισμένη -g=global και όχι τοπική. Αναφορικά με την εικόνα 7, εισάγω ως μεταβλητή το τρίτο όρισμα (var="\$@"), όπου \$@ παίρνει την τιμή του \$3 από την case της Εικόνας 1) που θα δωθεί στην κονσόλα (δηλαδή αν θέλω να μου εμφανίσει τις γραμμές μόνο συγκεκριμένου χρήστη), εισάγωντας την έτσι στην συνάρτηση `mining_usernames()`. Αν αυτή η μεταβλητή δεν δωθεί και επομένως "-z var" τότε η εντολή `awk` παίρνει την τρίτη στήλη του .log αρχείου μας, την εισάγει σε έναν πίνακα που αριθμεί πόσες φορές εμφανίζεται η συγκεκριμένη λέξη, και την εισάγει μέσω `pipe "|"` στην εντολή `sort` η οποία κατατάσσει τις εμφανίσεις με διαφορετική σειρά και ξανα με `pipe "|"` στην `uniq` ώστε να εμφανίσει μόνο τις ξεχωριστές εμφανίσεις και όχι και τις π.χ 124 εμφανίσεις του admin, χωρίς να πειράξει τις συνολικές εμφανίσεις που έχουν αποθηκευτεί στον πίνακα. Με την εκτέλεσή της παίρνουμε το παρακάτω αποτέλεσμα:

```
(kali@kali)-[~/test]
$ ./logparser.sh access.log --usrid
-          15710
admin      124
president  34
root       181
user1      110
user2      91
user3      39
```

Εικόνα 8: --usrid χωρίς τρίτο όρισμα

Στο δεύτερο σκέλος του υποερωτήματος, εφόσον δοθεί ποιανού χρήστη τα αναφορικά log θέλουμε να εκτυπώσουμε, η υλοποίησή του γίνεται με χρήση else και πάλι της awk, εισάγοντας το pat (εννοώντας pattern που αναζητάμε), ως το τρίτο όρισμα, και εκτυπώνοντας από το αρχείο μας μόνο τις γραμμές που αναφέρεται. Απόσπασμα από την εκτέλεση του:

```
(kali@kali)-[~/test]
$ ./logparser.sh access.log --usrid president
::1 - president - [10/Oct/2021:10:20:10 +0300] "GET /php
myadmin/themes/dot.gif HTTP/1.1" 200 43
::1 - president - [11/Oct/2021:10:20:10 +0300] "GET /php
myadmin/themes/pmahomme/img/b_docs.png HTTP/1.1" 200 705
::1 - president - [11/Oct/2021:10:20:10 +0300] "GET /php
myadmin/themes/pmahomme/img/b_import.png HTTP/1.1" 200 5
56
::1 - president - [14/Oct/2021:10:20:45 +0300] "GET /php
myadmin/themes/pmahomme/img/bd_empty.png HTTP/1.1" 200 3
27
::1 - president - [15/Oct/2021:10:22:23 +0300] "GET /php
myadmin/themes/pmahomme/img/b_insrow.png HTTP/1.1" 200 1
57
::1 - president - [16/Oct/2021:10:22:23 +0300] "GET /php
myadmin/themes/pmahomme/img/col_pointer.png HTTP/1.1" 20
0 102
::1 - president - [17/Oct/2021:10:22:43 +0300] "GET /MyD
ocs/AEKX/images/en.jpg HTTP/1.1" 200 759
::1 - president - [18/Oct/2021:10:30:23 +0300] "GET /php
myadmin/index.php?route=/sql&server=1&db=aekx&table=comp
liances&pos=0&ajax_request=true&ajax_page_request=true&_
nocache=1650353423706770348&token=32567d5d795a3d6d36285a
4b68702655 HTTP/1.1" 200 5325
::1 - president - [18/Oct/2021:13:40:32 +0300] "POST /ph
pmyadmin/index.php?route=/ HTTP/1.1" 200 1660
::1 - president - [28/Oct/2021:14:18:38 +0300] "GET /ico
ns/blank.gif HTTP/1.1" 200 148
::1 - president - [10/Nov/2021:14:18:49 +0300] "GET /MyD
ocs/AEKX/login.php HTTP/1.1" 302 -
```

Εικόνα 9: Εκτέλεση με τρίτο όρισμα "president"

Στην Εικόνα 9 μας εμφανίζονται απόσπασματα μόνο από τις γραμμές που ο president εμφανίζεται στην 3<sup>η</sup> στήλη του access.log. Αντίστοιχα λειτουργεί και για τον κενό χρήστη (-) και τους χρήστες admin, root, user1, user2, user3.

(iv) Εφόσον δίνεται ως δεύτερο όρισμα η εντολή '-method', ενεργοποιείται η αντίστοιχη case στην γραμμή 77 της Εικόνας 1 που περνάει ως όρισμα στην συνάρτηση method() το τρίτο όρισμα που θα δώσει ο χρήστης στον τερματικό ως εντολή, δηλαδή GET ή POST.

Η συνάρτηση method υλοποιείται ως εξής:

```
22 method(){
23 var="$@"
24 if [[ -z $var ]] || [[ $var ≠ "GET" ]] && [[ $var ≠
   "POST" ]]; then echo "Wrong Method Name"
25
26
27 else
28     awk -v pat="$var" '$0~pat' $filename
29
30 fi
31
32 }
```

Εικόνα 10: συνάρτηση method

Κατά τα γνωστά περνάμε την τιμή του τρίτου ορίσματος που θα δώσει ο χρήστης στην μεταβλητή var, και ελέγχουμε αν (1) Ο χρήστης έδωσε κενό όρισμα ή αν (2) δεν έδωσε το όρισμα GET ή αν (3) δεν έδωσε το όρισμα POST, που σε περίπτωση που ισχύει κάποια από αυτές τις προϋποθέσεις τότε το script εμφανίζει μήνυμα "Wrong Method Name", όπως υποδεικνύεται στην εκφώνηση. Εφόσον η συνθήκη αυτή δεν είναι έγκυρη



και ο χρήστης έχει εισαγάγει το ορθό όρισμα, αυτό περνάει στην εντολή awk και εκτυπώνονται μόνον οι γραμμές που περιέχουν GET ή POST:

```
(kali@kali)-[~/test]
$ ./logparser.sh access.log -method POST
127.0.0.1 - - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/config/get HTTP/1.1" 200 1563 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/config/get HTTP/1.1" 200 1654 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/navigation&ajax_request=1 HTTP/1.1" 200 2266 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/database/structure/favorite-table&ajax_request=1&favorite_table=1&sync_favorite_tables=1&lang=en HTTP/1.1" 200 155 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - user1 - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/version-check HTTP/1.1" 200 64 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [30/Mar/2021:16:47:25 +0300] "POST /phpmyadmin/index.php?route=/config/set HTTP/1.1" 200 1554 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - admin - [30/Mar/2021:16:47:42 +0300] "POST /phpmyadmin/index.php?route=/import HTTP/1.1" 200 9527 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
```

Εικόνα 11: εντολή -method με όρισμα POST (απόσπασμα)

Εκτέλεση με λάθος όρισμα:

```
(kali@kali)-[~/test]
$ ./logparser.sh access.log -method lathos
Wrong Method Name
```

(v) Εφόσον δίνεται ως δεύτερο όρισμα η εντολή '--serprot', ενεργοποιείται η αντίστοιχη case στην γραμμή 79 της Εικόνας 1 που περνάει ως όρισμα στην συνάρτηση getip() το τρίτο όρισμα που θα δώσει ο χρήστης στον τερματικό ως εντολή, δηλαδή IPv4 ή IPv6. Ο κώδικας της και η εκτέλεση της είναι παρόμοιος και βασισμένος σε αυτόν του υποερωτήματος (iv). Η μόνη διαφορά έγκειται στο ότι οι γραμμές που ταιριάζουν με το τρίτο όρισμα του χρήστη εκτυπώνονται με την εντολή sed καθαρά για λόγους υλοποίησης του project, καθώς η αρχική τους υλοποίηση είχε γίνει με την χρήση της awk.

```

45 getip(){
46 var="$@"
47 if [[ -z $var ]] || [[ $var ≠ "IPv4" ]] && [[ $var ≠
  "IPv6" ]]; then echo "Wrong Method Name"
48
49 elif [[ $var = "IPv4" ]]; then
50     sed -n '/127.0.0.1/p' $filename
51 else     sed -n '/::1/p' $filename
52 fi
53 }
54

```

Εικόνα 12: συνάρτηση getip() με χρήση sed

```

49 getip(){
50 var="$@"
51 if [[ -z $var ]] || [[ $var ≠ "IPv4" ]] && [[ $var ≠
  "IPv6" ]]; then echo "Wrong Method Name"
52 elif [[ $var = "IPv4" ]]; then
53     awk '/127.0.0.1/ {print}' $filename
54 else     awk '/::1/ {print}' $filename
55 fi
56 }

```

Εικόνα 13: συνάρτηση getip() με χρήση awk

```

(kali㉿kali)-[~/test]
$ ./logparser.sh access.log --servprot lathos
Wrong Method Name

```

Εικόνα 14: Εκτέλεση --servprot με λάθος όρισμα

```

(kali@kali)-[~/test]
$ ./logparser.sh access.log --servprot IPv4
127.0.0.1 - root - [29/Mar/2021:16:47:19 +0300] "GET / HTTP/1.1" 302 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [29/Mar/2021:16:47:19 +0300] "GET /dashboard/ HTTP/1.1" 200 7577 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - admin - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/stylesheets/normalize.css HTTP/1.1" 200 6876 "http://localhost/dashboard/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/stylesheets/all.css HTTP/1.1" 200 481698 "http://localhost/dashboard/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/javascripts/modernizr.js HTTP/1.1" 200 51365 "http://localhost/dashboard/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - user1 - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/javascripts/all.js HTTP/1.1" 200 188385 "http://localhost/dashboard/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"
127.0.0.1 - - [29/Mar/2021:16:47:20 +0300] "GET /dashboard/images/bitnami-xampp.png HTTP/1.1" 200 22133 "http://localhost/dashboard/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0"

```

Εικόνα 15: Απόσπασμα εκτέλεσης με IPv4 όρισμα. Εκτυπώνονται μόνο τα πεδία με 127.0.0.1

(vi) Εφόσον δίνεται ως δεύτερο όρισμα η εντολή ‘—browsers’, ενεργοποιείται η αντίστοιχη case στην γραμμή 78 της Εικόνας 1 που περνάει ως όρισμα στην συνάρτηση count\_browsers(), αυτή την φορά χωρίς τρίτο όρισμα, αφού δεν απαιτείται από την εκφώνηση. Στην count\_browsers() εμπεριέχεται μια συνάρτηση match(). Δεν αντιμετωπίσα κάποιο πρόβλημα εδώ αφού στην shell script μπορεί να καλείται συνάρτηση μέσα σε συνάρτηση.

```

34 count_browsers(){
35 match(){
36 for browser in "Mozilla" "Chrome" "Safari" "Edg"
37 do
38 awk -v pat=$browser 'BEGIN{count=0} $0~pat {count+=1}
   END {print pat "\t" count}' $filename
39 done
40 }
41 match
42 }

```

Εικόνα 16: Η συνάρτηση count\_browsers

Υλοποίησα την απαίτηση της εκφώνησης χρησιμοποιώντας έναν βρόχο for προκειμένου να εισάγω τις 4 διαφορετικές λέξεις που ψάχνουμε, εισάγοντας μετά την λίστα browser ως μεταβλητή pat στην awk και προσθέτοντας +1 στην μεταβλητή count κάθε φορά που εμφανιζόταν κάποια από τις λέξεις στο log αρχείο μας .

```
(kali㉿kali)-[~/test]
$ ./logparser.sh access.log --browsers
Mozilla 1057
Chrome 124
Safari 124
Edg 63
```

Εικόνα 17: Εκτέλεση της --browsers

(vii) Εφόσον δίνεται ως δεύτερο όρισμα η εντολή '—datum' , ενεργοποιείται η αντίστοιχη case στην γραμμή 80 της Εικόνας 1 που περνάει ως όρισμα στην συνάρτηση getdate() το τρίτο όρισμα που θα δώσει ο χρήστης στον τερματικό ως εντολή, δηλαδή ποιον μήνα επιθυμεί.

```
55 getdate(){
56 var="$@"
57 month="Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec"
58
59 if [[ -z $var ]]; then echo "Wrong Date"
60 elif grep -q -w $var <<< "$month" ; then awk -v
    pat="$var" '$0~pat' $filename
61 else echo "Wrong Date"
62 fi
63 }
64
```

Εικόνα 18: Συνάρτηση getdate()

Δημιούργησα μια μεταβλητή και κωδικοποίησα τις δυνατές καταστάσεις δηλαδή τις δυνατές τιμές του τρίτου ορίσματος (τους μήνες).

Περνάμε τις λέξεις που περιέχονται στην month στην εντολή grep, την καθεμία ξεχωριστά μέσω της παραμέτρου -w και χρησιμοποιούμε την -q προκειμένου να μην εμφανίσει στον χρήστη όταν εκτελέσει την εντολή : «Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec». Οι λέξεις αυτές περνάνε στην pat μεταβλητή και σαρώνουν το κείμενο εμφανίζοντας τις γραμμές εκείνες που εμφανίζεται το τρίτο όρισμα εφόσον είναι κάποιο από την λίστα month. Σε κάθε άλλη περίπτωση εμφανίζεται το μήνυμα «Wrong Date» σύμφωνα με τις υποδείξεις της εκφώνησης.

Το συγκεκριμένο ερώτημα μου φάνηκε λίγο πιο δύσκολο από τα υπόλοιπα. Η ιδέα της υλοποίησης είναι δικιά μου (χρήση month=".."), ωστόσο χρησιμοποίησα αρκετή βοήθεια από το διαδίκτυο (κυρίως stack overflow), προκειμένου να συντάξω ορθά την grep. Με μπέρδεψε που γινόταν η χρήση μονού - διπλού εισαγωγικού καθώς και η τριπλή εκχώρηση.



```
(kali㉿kali)-[~/test]
$ ./logparser.sh access.log --datum
Wrong Date
```

Εικόνα 19: Λανθασμένο όρισμα: κενό όρισμα

```
(kali㉿kali)-[~/test]
$ ./logparser.sh access.log --datum jan
Wrong Date
```

Εικόνα 20: Μη προσδιορισμένο στην λίστα Month όρισμα

```
(kali㉿kali)-[~/test]
$ ./logparser.sh access.log --datum Jan
:: 1 - - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/AEKX/
HTTP/1.1" 200 5328
:: 1 - - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/AEKX/s
cripts/jquery-1.9.1.min.js HTTP/1.1" 200 92596
:: 1 - user1 - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/
AEKX/css/bootstrap.min.css HTTP/1.1" 200 163873
:: 1 - admin - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/
AEKX/css/dashboard.css HTTP/1.1" 200 1573
:: 1 - - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/AEKX/i
mages/oke2.svg HTTP/1.1" 200 2441
:: 1 - user2 - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/
AEKX/images/en.jpg HTTP/1.1" 200 759
:: 1 - - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/AEKX/i
mages/logo.jpg HTTP/1.1" 200 55235
:: 1 - root - [14/Jan/2022:14:26:15 +0300] "GET /MyDocs/A
EKX/images/en_on.jpg HTTP/1.1" 200 661
:: 1 - - [14/Jan/2022:16:29:25 +0300] "GET /MyDocs/AEKX/
```

Εικόνα 21: Ορθή χρήση του τρίτου ορίσματος (απόσπασμα)

Συνολικά ο κώδικας του αρχείου logparser.sh :

```
#!/bin/bash
#1084537
#1084538
#1084522

declare -g filename=$1

id(){
    echo "1084537|1084538|1084522"
}
```

```

mining_usernames(){
var="$@"
if [[ -z $var ]]; then awk ' { arr[$3]++ } END { for( no in arr) { printf no "\t"
arr[no] "\n"} } ' $filename |sort | uniq
else awk -v pat="-- $var" '$0~pat' $filename
fi
}

method(){
var="$@"
if [[ -z $var ]] || [[ $var != "GET" ]] && [[ $var != "POST" ]]; then echo "Wrong
Method Name"

else
    awk -v pat="$var" '$0~pat' $filename
fi
}

count_browsers(){
match(){
for browser in "Mozilla" "Chrome" "Safari" "Edg"
do
awk -v pat=$browser 'BEGIN{count=0} $0~pat {count+=1} END {print pat "\t"
count}' $filename
done
}
match
}

getip(){
var="$@"
if [[ -z $var ]] || [[ $var != "IPv4" ]] && [[ $var != "IPv6" ]]; then echo
"Wrong Method Name"

elif [[ $var == "IPv4" ]]; then
    sed -n '/127.0.0.1/p' $filename
else sed -n '/::1/p' $filename
fi
}

getdate(){
var="$@"
month="Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec"

if [[ -z $var ]]; then echo "Wrong Date"
elif grep -q -w $var <<< "$month" ; then awk -v pat="$var" '$0~pat' $filename

```

```

else echo "Wrong Date"
fi
}

if [[ 0 == $# ]]; then
    id

elif [[ $filename == *.log ]]; then
    opt="$2";
    case "$opt" in
        "--usrid" ) mining_usernames "$3" ;;
        "--method" ) method "$3" ;;
        "--browsers" ) count browsers ;;
        "--servprot" ) getip "$3" ;;
        "--datum" ) getdate "$3" ;;
        "" ) cat -n $1 ;;
        * ) echo "Wrong argument \o/ " ;;
    esac
elif [[ $filename != *.log ]]; then
    echo "Wrong File Argument"
fi

```

## ΕΡΩΤΗΜΑ 2: ΔΙΕΡΓΑΣΙΕΣ

Ο κώδικας είναι μια διαδοχική υλοποίηση αριθμητικής ολοκλήρωσης χρησιμοποιώντας τον τραπεζοειδή κανόνα. Η συνάρτηση που ενσωματώνεται ορίζεται από τη συνάρτηση "f" και το διάστημα ολοκλήρωσης είναι [1, 4]. Ο κώδικας υπολογίζει την ακέραια τιμή διαιρώντας το διάστημα σε υποδιαστήματα "n", καθένα πλάτους "dx", και προσεγγίζοντας τη συνάρτηση στο μέσο κάθε υποδιαστήματος.

Ο κώδικας έχει επίσης μερικές προσθήκες για την υλοποίηση παράλληλης επεξεργασίας χρησιμοποιώντας τη συνάρτηση fork() για τη δημιουργία διεργασιών παιδιών. Κάθε διεργασία παιδί υπολογίζει την ακέραια τιμή για ένα τμήμα των υποδιαστημάτων και τα αποτελέσματα αποστέλλονται σε μια ουρά μηνυμάτων για να συλλέξει και να συνοψίσει η γονική διαδικασία. Η ουρά μηνυμάτων δημιουργείται χρησιμοποιώντας τη βιβλιοθήκη mqueue.h, η οποία παρέχει λειτουργικότητα για τη δημιουργία και τον χειρισμό ουρών μηνυμάτων στο Linux. Η γονική διαδικασία περιμένει να ολοκληρωθούν όλες οι διεργασίες παιδιά χρησιμοποιώντας τη συνάρτηση wait().

Ο κώδικας μετρά τον χρόνο εκτέλεσης χρησιμοποιώντας τη συνάρτηση gettimeofday() από τη βιβλιοθήκη sys/time.h και αναφέρει το αποτέλεσμα και τον χρόνο εκτέλεσης στην κονσόλα.

Προκειμένου να υλοποιηθεί ο κώδικας, απαιτείται η εισαγωγή των παρακάτω βιβλιοθηκών. Συγκεκριμένα, η <sys/time.h> δίνεται από την εκφώνηση και είναι βιβλιοθήκη με λειτουργίες στο λειτουργικό unix.

```
// integral_seq.c numerical integration - sequential code
#include <stdio.h>
#include <math.h>
#include <sys/time.h>
```

Παράλληλα προσθέτουμε τις παρακάτω βιβλιοθηκές. Η <sys/wait.h> είναι επίσης στο λειτουργικό unix και είναι απαραίτητη για την υλοποίηση των ζητούμενων. Παρουσιάζουμε:

```
//prosthiki
#include <mqueue.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
//telos prosthikis
```

Ο κώδικας που μας δίνεται υπολογίζει το ολοκλήρωμα της συνάρτησης  $f(x) = \ln(x) * \sqrt{x}$  στο διάστημα [1, 4]. Βασικός στόχος μας είναι η επικοινωνία των διαδικασιών παιδιών μεταξύ τους ώστε να υπολογιστεί το ολοκλήρωμα.

Δημιουργούμε την απαραίτητη ουρά μηνυμάτων που λειτουργεί ως εξής :

Ο αριθμός των διεργασιών που θα δημιουργηθούν μεταβιβάζεται ως το πρώτο όρισμα στο πρόγραμμα. Τα ακόλουθα βήματα εκτελούνται σε αυτό το απόσπασμα κώδικα:



num\_procs: Ο αριθμός των διεργασιών που θα δημιουργηθούν αποθηκεύεται στο num\_procs μετά τη μετατροπή του ορίσματος συμβολοσειράς argv[1] σε ακέραιο χρησιμοποιώντας τη συνάρτηση atoi().

Χαρακτηριστικά ουράς μηνυμάτων: Ορίζεται μια δομή attr, η οποία περιλαμβάνει:

mq\_flags: Καθορίζει τις σημαίες για την ουρά μηνυμάτων. Η τιμή 0 σημαίνει ότι δεν έχουν οριστεί σημαίες.

mq\_maxmsg: Καθορίζει τον μέγιστο αριθμό μηνυμάτων που επιτρέπεται στην ουρά.

mq\_msgsize: Καθορίζει το μέγιστο μέγεθος κάθε μηνύματος στην ουρά.

mq\_curmsgs: Καθορίζει τον τρέχοντα αριθμό μηνυμάτων στην ουρά.

Δημιουργία ουράς μηνυμάτων: Η ουρά μηνυμάτων δημιουργείται χρησιμοποιώντας τη συνάρτηση mq\_open() με το όνομα "/integral\_queue", με σημαίες O\_CREAT, O\_RDWR (δημιουργία και άνοιγμα για ανάγνωση και εγγραφή), και τα καθορισμένα χαρακτηριστικά ουράς μηνυμάτων. Η επιστρεφόμενη τιμή από mq\_open() αποθηκεύεται στον περιγραφέα ουράς μηνυμάτων mq.

Δημιουργία διαδικασίας: Ένας βρόχος for χρησιμοποιείται για τη δημιουργία διεργασιών num\_procs χρησιμοποιώντας τη συνάρτηση fork(). Το αναγνωριστικό διεργασίας αποθηκεύεται στο pid. Εάν το pid είναι 0, σημαίνει ότι η τρέχουσα διαδικασία είναι παιδί.

Child Process: Στη διαδικασία παιδί υπολογίζεται η αρχή και το τέλος, τα οποία καθορίζουν το εύρος του βρόχου στον οποίο η διαδικασία παιδί θα εκτελέσει τους υπολογισμούς της. Το child\_S είναι μια μεταβλητή για την αποθήκευση του αποτελέσματος της διαδικασίας παιδί. Ο βρόχος υπολογίζει το παιδί\_S χρησιμοποιώντας τη συνάρτηση f() που ορίστηκε νωρίτερα. Τέλος, το child\_S πολλαπλασιάζεται με dx και αποστέλλεται στη γονική διαδικασία χρησιμοποιώντας τη συνάρτηση mq\_send(). Στη συνέχεια, η διαδικασία παιδί κλείνει την ουρά μηνυμάτων και εξέρχεται.

Γονική διαδικασία: περιμένει όλες τις διεργασίες παιδιά να ολοκληρώσουν την εργασία τους και να στείλουν τα αποτελέσματά τους στην ουρά μηνυμάτων. Επιπλέον, λαμβάνει τα αποτελέσματα χρησιμοποιώντας mq\_receive() και τα προσθέτει στη μεταβλητή S. Αφού ληφθούν όλα τα αποτελέσματα, κλείνει και αποσυνδέει την ουρά μηνυμάτων και περιμένει την έξοδο των διεργασιών παιδιών.

Αυτό το τμήμα του προγράμματος εκτελείται, αφού δημιουργήσει num\_procs διεργασίες παιδιά.

Ουρά:

```
//prosthiki
int num_procs = atoi(argv[1]);
mqd_t mq;
struct mq_attr attr;
attr.mq_flags = 0;//Kathorismos shmaiwn
attr.mq_maxmsg = 10;//Megistos arithmos mhn
attr.mq_msgsize = sizeof(double);//Megisto megethos mhnymatos
attr.mq_curmsgs = 0;//Trexwn arithmos
mq = mq_open("/integral_queue", O_CREAT | O_RDWR, 0644, &attr);
pid_t pid;
//FOR YLOPOIHSH
for (int i = 0; i < num_procs; i++) {
    pid = fork();
    if (pid == 0) {
        // child process
```

```

unsigned long start = i * (n / num_procs);
unsigned long end = (i + 1) * (n / num_procs);
double child_S = 0;
//telos prosthikis

    for (unsigned long i = start; i < end; i++) {
        double xi = a + (i + 0.5)*dx;
        child_S += f(xi);
    }
child_S *= dx;
//telos prosthikis

```

#### **Next Part:**

Η γονική διαδικασία χρησιμοποιεί τη συνάρτηση `mq_receive` για να λαμβάνει μηνύματα από μια ουρά μηνυμάτων με το όνομα `"/integral_queue"`. Αυτή η ουρά μηνυμάτων δημιουργείται νωρίτερα στον κώδικα με `mq_open` και χρησιμοποιείται για την αποστολή των αποτελεσμάτων των διεργασιών παιδιών στη γονική διαδικασία.

Η συνάρτηση `mq_receive` παίρνει τέσσερα ορίσματα:

**mq:** Ο περιγραφέας ουράς μηνυμάτων, που δημιουργήθηκε και άνοιξε με `mq_open`.

**(char \*) &received\_S:** Ένας δείκτης σε μια μεταβλητή τύπου `double` όπου θα αποθηκευτεί το ληφθέν μήνυμα.

**sizeof(double):** Το μέγεθος του ληφθέντος μηνύματος σε `byte`.

**0:** Η προτεραιότητα του μηνύματος.

Η γονική διεργασία επαναφέρει `'num_procs'` φορές και χρησιμοποιεί `mq_receive` για να λαμβάνει μηνύματα `num_procs`, καθένα από τα οποία αντιπροσωπεύει το αποτέλεσμα μιας από τις διεργασίες παιδιά. Το αποτέλεσμα κάθε διαδικασίας παιδί αποθηκεύεται στη μεταβλητή `receive_S` και προστίθεται στη μεταβλητή `S`, η οποία συγκεντρώνει το άθροισμα όλων των αποτελεσμάτων από τις διεργασίες παιδιά.

Στο τέλος, η μεταβλητή `S` θα περιέχει το τελικό αποτέλεσμα του υπολογισμού της αριθμητικής ολοκλήρωσης, που είναι το άθροισμα των αποτελεσμάτων κάθε διαδικασίας παιδιού.

```

//prosthiki
mq_send(mq, (char *) &child_S, sizeof(double),
0); //perigrafeas
mq_close(mq);
exit(0);

```

Το παρακάτω τμήμα του κώδικα εκτελεί τα ακόλουθα βήματα:

Κλείσιμο της ουράς μηνυμάτων: Η συνάρτηση `mq_close` χρησιμοποιείται για το κλείσιμο της που είχε ανοίξει προηγουμένως χρησιμοποιώντας το `mq_open`.

Αποσύνδεση της ουράς μηνυμάτων: Η συνάρτηση `mq_unlink` χρησιμοποιείται για την αποσύνδεση ή την αφαίρεσή της από το σύστημα.

Αναμονή για τον τερματισμό των διεργασιών παιδιών: Η συνάρτηση αναμονής χρησιμοποιείται για την αναμονή για τον τερματισμό όλων των διεργασιών παιδιών. Το όρισμα που μεταβιβάστηκε στην αναμονή είναι `NULL`, που σημαίνει ότι η συνάρτηση θα περιμένει να τερματιστεί οποιαδήποτε διαδικασία παιδί. Ο σκοπός της αναμονής για τον τερματισμό των διεργασιών παιδιών είναι να διασφαλιστεί ότι όλοι οι πόροι που χρησιμοποιούνται από αυτές ελευθερώνονται πριν από τον τερματισμό της γονικής διαδικασίας.

```
//prosthiki
mq_send(mq, (char *) &child_S, sizeof(double), 0);
mq_close(mq);
exit(0);
```

Με το συνδυασμό και την υλοποίηση των πληροφοριών που περιγράψαμε καταφέρνουμε να επικοινωνούν οι διάφορες διεργασίες μεταξύ τους και να έχουμε ένα επιθυμητό αποτέλεσμα .Τέλος , παραθέτουμε ολόκληρο τον κώδικα :

```
// integral_seq.c numerical integration - sequential code
#include <stdio.h>
#include <math.h>
#include <sys/time.h>

//prosthiki
#include <mqueue.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
//telos prosthikis

double get_wtime(void)
{
    struct timeval t;
    gettimeofday(&t, NULL);
    return (double)t.tv_sec + (double)t.tv_usec*1.0e-6;
}

double f(double x)
{
    return log(x)*sqrt(x);
}

// WolframAlpha: integral_1^4 log(x) sqrt(x) dx = 4/9 (4 log(64)-7)
// -> 4.28245881486164
int main(int argc, char *argv[])
{
```

```

double a = 1.0;
double b = 4.0;
unsigned long const n = 1e9;
const double dx = (b-a)/n;

double S = 0;

double t0 = get_wtime();

//prosthiki
int num_procs = atoi(argv[1]);
mqd_t mq;
struct mq_attr attr;
attr.mq_flags = 0;
attr.mq_maxmsg = 10;
attr.mq_msgsize = sizeof(double);
attr.mq_curmsgs = 0;
mq = mq_open("/integral_queue", O_CREAT | O_RDWR, 0644, &attr);
pid_t pid;

for (int i = 0; i < num_procs; i++) {
    pid = fork();
    if (pid == 0) {
        // child process
        unsigned long start = i * (n / num_procs);
        unsigned long end = (i + 1) * (n / num_procs);
        double child_S = 0;
        //telos prosthikis

        for (unsigned long i = start; i < end; i++) {
            double xi = a + (i + 0.5)*dx;
            child_S += f(xi);
        }
        child_S *= dx;
        //telos prosthikis

        //prosthiki
        mq_send(mq, (char *) &child_S, sizeof(double), 0);
        mq_close(mq);
        exit(0);
    }
}

// parent process
double received_S;
for (int i = 0; i < num_procs; i++) {
    mq_receive(mq, (char *) &received_S, sizeof(double), 0);
    S += received_S;
}

```



```

//telos prosthikis

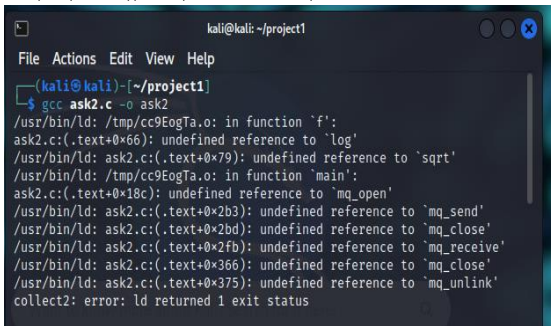
double t1 = get_wtime();
printf("Time=%lf seconds, Result=%.8f\n", t1-t0, S);

//prosthiki
mq_close(mq);
mq_unlink("/integral_queue");
for (int i = 0; i < num_procs; i++) {
    wait(NULL);
}
return 0;
}

```

### **ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ ΣΤΟ VIRTUAL MACHINE:**

- Ανοίγουμε και τρέχουμε το Virtual BOX
- Έχουμε επιλέξει να εγκαταστήσουμε το Virtual Machine Kali-Linux-2022.3-VirtualBox-amd64
- Αποθηκεύουμε ένα αρχείο με όνομα ask2 με σκοπό να παρουσιάσουμε το αποτέλεσμα του προγράμματος.
- Προσπαθούμε αρχικά να κάνουμε ένα απλό compile με την εντολή `gcc ask2.c -o ask2`, ωστόσο παρατηρούμε το παρακάτω error:

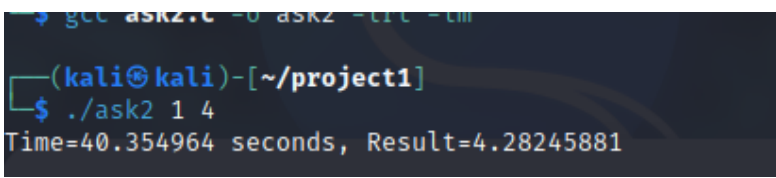


```

kali@kali: ~/project1
File Actions Edit View Help
(kali@kali)-[~/project1]
└─$ gcc ask2.c -o ask2
/usr/bin/ld: /tmp/cc9EogTa.o: in function 'f':
ask2.c:(.text+0x66): undefined reference to 'log'
/usr/bin/ld: ask2.c:(.text+0x79): undefined reference to 'sqrt'
/usr/bin/ld: /tmp/cc9EogTa.o: in function 'main':
ask2.c:(.text+0x18c): undefined reference to 'mq_open'
/usr/bin/ld: ask2.c:(.text+0x2b3): undefined reference to 'mq_send'
/usr/bin/ld: ask2.c:(.text+0x2bd): undefined reference to 'mq_close'
/usr/bin/ld: ask2.c:(.text+0x2fb): undefined reference to 'mq_receive'
/usr/bin/ld: ask2.c:(.text+0x366): undefined reference to 'mq_close'
/usr/bin/ld: ask2.c:(.text+0x375): undefined reference to 'mq_unlink'
collect2: error: ld returned 1 exit status

```

- Στη συνέχεια τρέχουμε την παρακάτω εντολή την οποία βρήκαμε από forum(stack overflow) και μπορέσαμε να τρέξουμε κανονικά το αρχείο και να βρούμε την λύση που φαίνεται και στον κώδικα (`// -> 4.2824588148616400`):



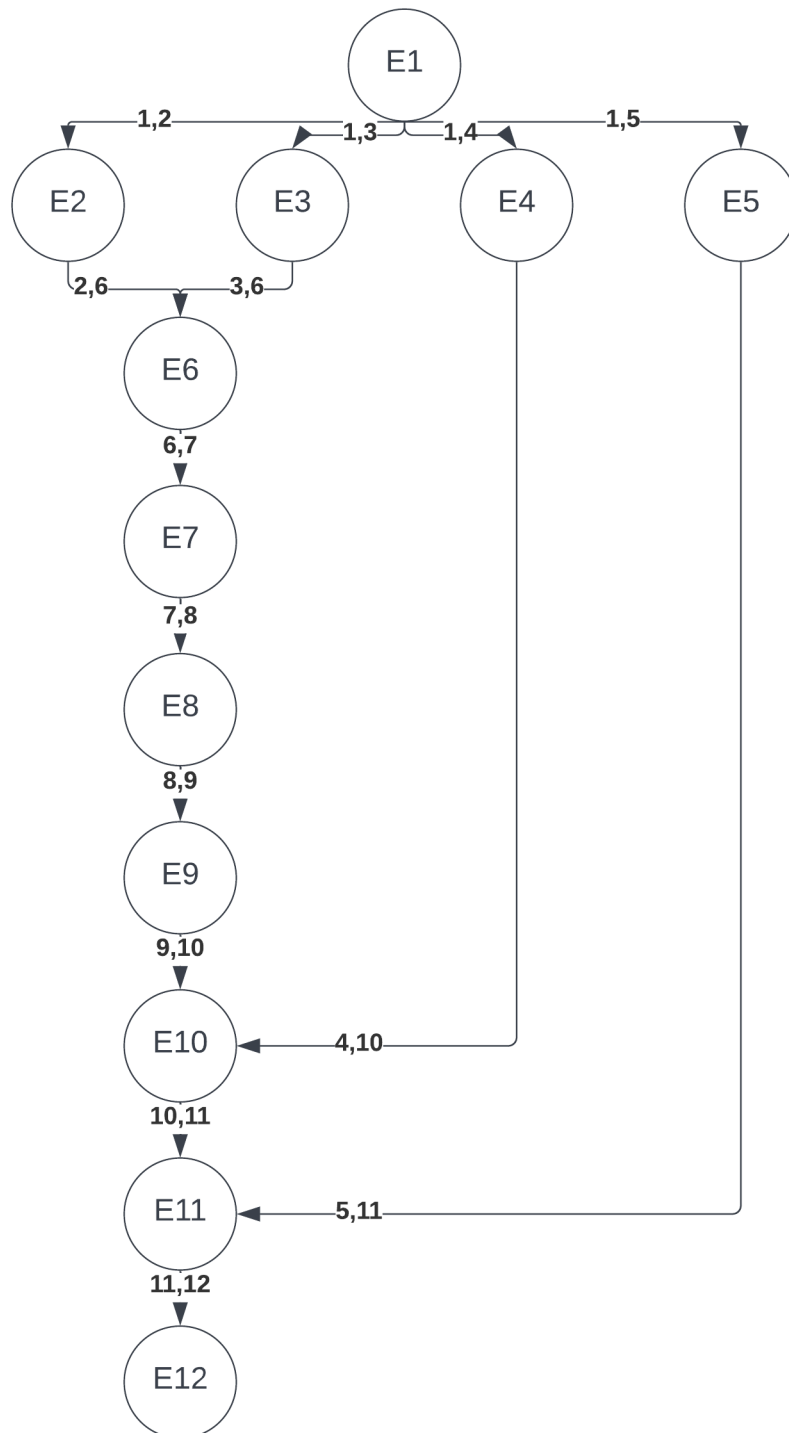
```

└─$ gcc ask2.c -o ask2 -lrt -lm
(kali@kali)-[~/project1]
└─$ ./ask2 1 4
Time=40.354964 seconds, Result=4.28245881

```

### **ΕΡΩΤΗΜΑ 3: ΔΙΑΔΙΕΡΓΑΣΙΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ**

(α)



(β)

```
BEGIN
E1;
END;
BEGIN
COBEGIN
E2;
E3;
E4;
E5;
COEND;
E6;
E7;
E8;
E9;
E10;
E11;
E12;
END;
```

(γ)

```
var Σ12, Σ13, Σ14, Σ15, Σ26, Σ36, Σ410, Σ511, Σ67, Σ78, Σ89, Σ910, Σ1011,
Σ1112:semaphores;
Σ12=Σ13=Σ14=Σ15=Σ26=Σ36=Σ410=Σ511=Σ67=Σ78=Σ89=Σ910=Σ1011=Σ1112=0;
cobegin
begin E1; signal(Σ12); signal(Σ13); signal(Σ14); signal(Σ15); end;
begin wait(Σ12); E2; signal(Σ26); end;
begin wait(Σ13); E3; signal(Σ36); end;
begin wait(Σ14); E4; signal(Σ410); end;
begin wait(Σ15); E5; signal(Σ511); end;
begin wait(Σ36); wait(Σ26); E6; signal(Σ67); end;
begin wait(Σ78); E7; signal(Σ89); end;
begin wait(Σ89); E9; signal(Σ910); end;
begin wait(Σ410); wait(Σ910); E10; signal(Σ1011); end;
begin wait(Σ1011); wait(Σ511); E11; signal(Σ1112); end;
begin wait(Σ1112); E12; end;
coend
```

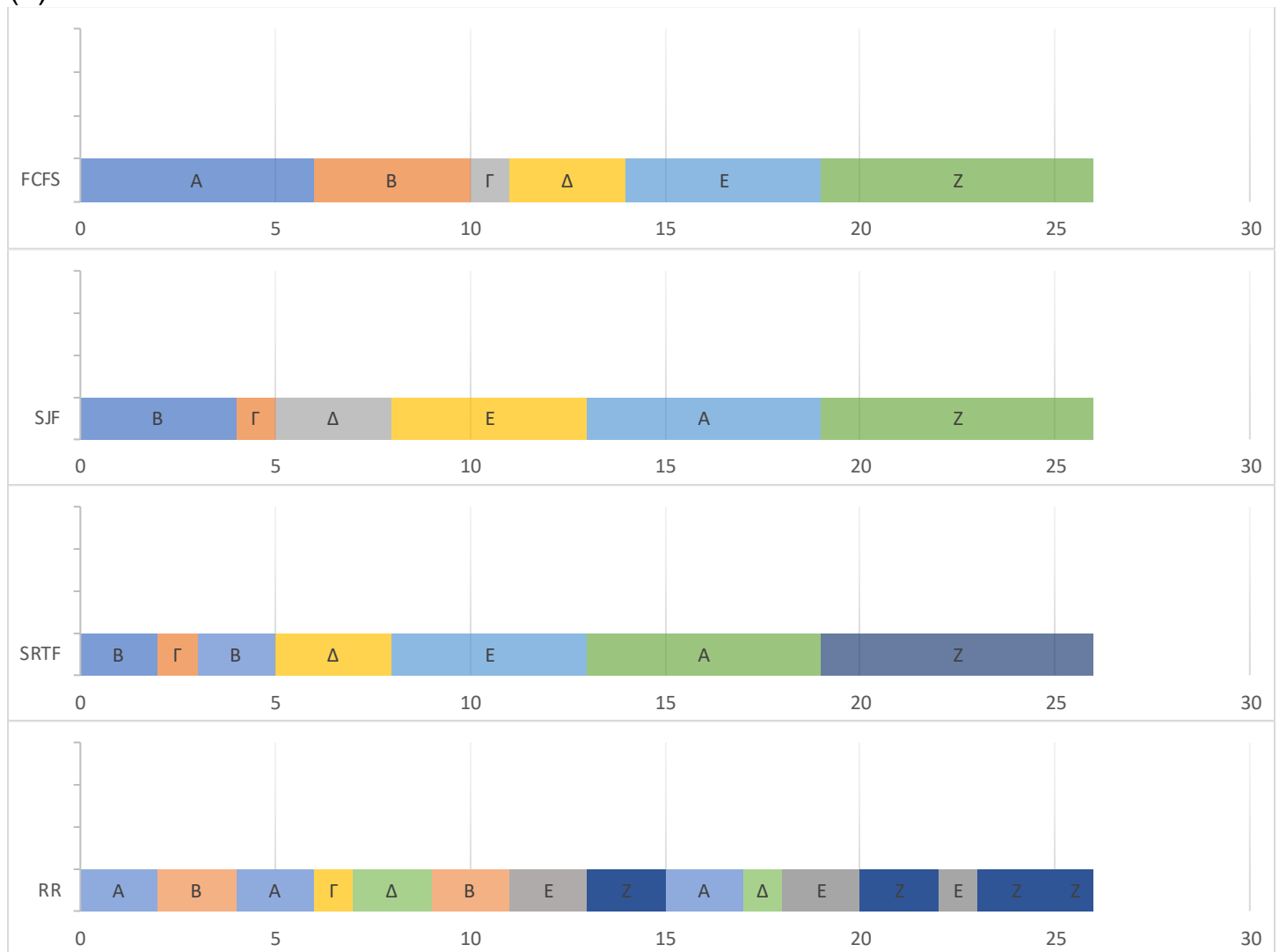
(8)

```
var  $\Sigma$ 1,  $\Sigma$ 2,  $\Sigma$ 3,  $\Sigma$ 4,  $\Sigma$ 5,  $\Sigma$ 6:semaphores;  
var  $\Sigma$ 1= $\Sigma$ 2= $\Sigma$ 3= $\Sigma$ 4= $\Sigma$ 5= $\Sigma$ 6=0;  
cobegin  
begin E1; signal( $\Sigma$ 1); end;  
begin wait( $\Sigma$ 1); E2; signal( $\Sigma$ 2); end;  
begin wait( $\Sigma$ 1); E3; signal( $\Sigma$ 3); end;  
begin wait( $\Sigma$ 1); E4; signal( $\Sigma$ 4); end;  
begin wait( $\Sigma$ 1); E5; signal( $\Sigma$ 5); end;  
begin wait( $\Sigma$ 2); wait( $\Sigma$ 3); E6; signal( $\Sigma$ 6); end;  
begin wait( $\Sigma$ 6); E7; end;  
begin wait( $\Sigma$ 6); E9; end;  
begin wait( $\Sigma$ 6); wait( $\Sigma$ 4); E10; end;  
begin wait( $\Sigma$ 6); wait( $\Sigma$ 5); E11; end;  
begin wait( $\Sigma$ 6); E12; end;  
coend
```



## ΕΡΩΤΗΜΑ 3: ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

(α)



(β)

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ: ΧΟ - Χαφ**

| <i>ΔΙΕΡΓΑΣΙΑ</i>     | <b>FCFS</b> | <b>SJF</b> | <b>SRTF</b> | <b>RR</b>  |
|----------------------|-------------|------------|-------------|------------|
| A                    | 6-0=6       | 19-0=19    | 19-0=19     | 17-0=17    |
| B                    | 10-0=10     | 4-0=4      | 5-0=5       | 11-0=11    |
| Γ                    | 11-2=9      | 5-2=3      | 3-2=1       | 7-2=5      |
| Δ                    | 14-3=11     | 8-3=5      | 8-3=5       | 18-3=15    |
| E                    | 19-4=15     | 13-4=9     | 13-4=9      | 23-4=19    |
| Z                    | 26-5=21     | 26-5=21    | 26-5=21     | 26-5=21    |
| <i>ΜΧΟΛΟΚΛΗΡΩΣΗΣ</i> | 72/6=12     | 61/6=10,16 | 60/6=6      | 86/6=14,33 |

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ: ΧΔ - ΧΕ**

| <b>ΔΙΕΡΓΑΣΙΑ</b>  | <b>FCFS</b> | <b>SJF</b> | <b>SRTF</b> | <b>RR</b> |
|-------------------|-------------|------------|-------------|-----------|
| A                 | 6-6=0       | 19-6=13    | 19-6=13     | 17-6=11   |
| B                 | 10-4=6      | 4-4=0      | 5-4=1       | 11-4=7    |
| Γ                 | 9-1=8       | 3-1=2      | 1-1=0       | 5-1=4     |
| Δ                 | 11-3=8      | 5-3=2      | 5-3=2       | 15-3=12   |
| E                 | 15-5=10     | 9-5=4      | 9-5=4       | 19-5=14   |
| Z                 | 21-7=14     | 21-7=14    | 21-7=14     | 19-7=13   |
| <i>ΜΧΑΝΑΜΟΝΗΣ</i> | 46/6=7,66   | 35/6=5,8   | 34/6=5,6    | 66/6=11   |

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ: Χαρχ - Χαφ**

| <b>ΔΙΕΡΓΑΣΙΑ</b>   | <b>FCFS</b> | <b>SJF</b> | <b>SRTF</b> | <b>RR</b> |
|--------------------|-------------|------------|-------------|-----------|
| A                  | 0-0=0       | 13-0=13    | 13-0=13     | 0-0=0     |
| B                  | 10-4=6      | 0-0=0      | 0-0=0       | 2-0=2     |
| Γ                  | 10-2=8      | 4-2=2      | 2-2=0       | 6-2=4     |
| Δ                  | 11-3=8      | 5-3=2      | 5-3=2       | 7-3=4     |
| E                  | 14-4=10     | 8-4=4      | 8-4=4       | 11-4=7    |
| Z                  | 19-5=14     | 19-5=14    | 19-5=14     | 13-5=8    |
| <i>ΜΧΑΠΟΚΡΙΣΗΣ</i> | 46/6=7,66   | 35/6=5,8   | 33/6=5,5    | 25/6=4,1  |

**ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ: ΟΤΑΝ ΔΙΑΚΟΠΤΕΤΑΙ Η ΔΙΕΡΓΑΣΙΑ**

| <b>ΔΙΕΡΓΑΣΙΑ</b> | <b>FCFS</b> | <b>SJF</b> | <b>SRTF</b> | <b>RR</b> |
|------------------|-------------|------------|-------------|-----------|
| A                | 0           | 0          | 0           | 2         |
| B                | 0           | 0          | 1           | 1         |
| Γ                | 0           | 0          | 0           | 0         |
| Δ                | 0           | 0          | 0           | 1         |
| E                | 0           | 0          | 0           | 2         |
| Z                | 0           | 0          | 0           | 3         |

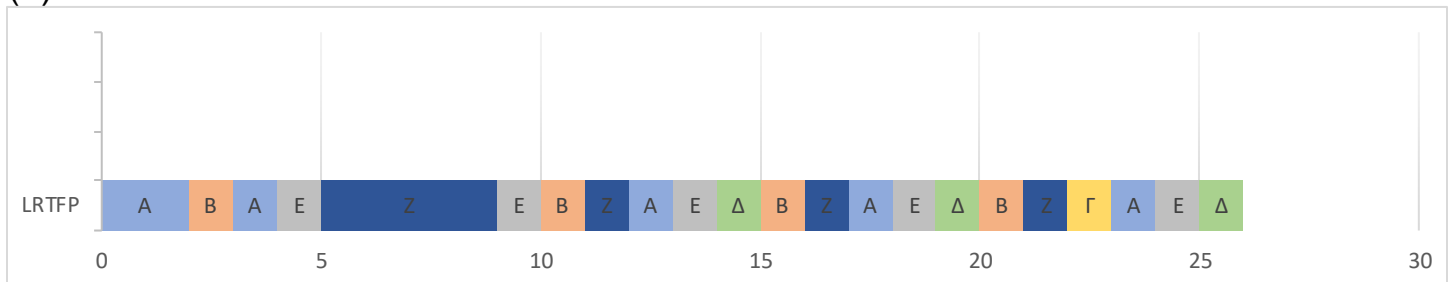
(Υ)

$$\frac{FCFS - SJF}{FCFS} * 100 = 15,8\%$$

$$\frac{FCFS - RR}{FCFS} * 100 = -19,1\%$$

$$\frac{FCFS - SRTF}{FCFS} * 100 = 50\%$$

(δ)



| ΣΤΙΓΜΗ | ΘΑ ΤΡΕΞΕΙ | ΧΡΟΝΟΣ                                     | ΣΧΟΛΙΟ |
|--------|-----------|--|--------|
| 0      | A         | $A > B$                                    | LR     |
| 1      | A         | $A > B$                                    | LR     |
| 2      | A         | $A = B > \Gamma$                           | PID    |
| 3      | B         | $A > \Delta = B > \Gamma$                  | LR     |
| 4      | A         | $E > B, A, \Delta, \Gamma$                 | LR     |
| 5      | E         | $Z > E, B, A, \Delta, \Gamma$              | LR     |
| 6      | Z         | $Z > E, B, A, \Delta, \Gamma$              | LR     |
| 7      | Z         | $Z = E$ ΑΛΛΑ PID                           | LR     |
| 8      | Z         | $E > B, A, \Delta, \Gamma$                 | PID    |
| 9      | E         | $Z = B$ ΑΛΛΑ PID                           | LR     |
| 10     | B         | $Z = A = \Delta = \Gamma$ ΑΛΛΑ PID         | PID    |
| 11     | Z         | $A = \Delta = \Gamma$ ΑΛΛΑ PID             | PID    |
| 12     | A         | $E = \Gamma = \Delta$ ΑΛΛΑ PID             | PID    |
| 13     | E         | $\Delta = \Gamma$ ΑΛΛΑ PID                 | PID    |
| 14     | Δ         | $B > \Delta, Z, E, \Gamma, A$              | PID    |
| 15     | B         | $\Delta = Z = E = \Gamma = A = B$ ΑΛΛΑ PID | PID    |
| 16     | Z         | $\Delta = Z = E = \Gamma = A$ ΑΛΛΑ PID     | PID    |
| 17     | A         | $\Delta = E = \Gamma = A$ ΑΛΛΑ PID         | PID    |
| 18     | E         | $\Delta > \Gamma$ ΑΛΛΑ PID                 | PID    |
| 19     | Δ         | $\Delta = Z = E = \Gamma = A = B$ ΑΛΛΑ PID | PID    |
| 20     | B         | $\Delta = Z = E = \Gamma = A$ ΑΛΛΑ PID     | PID    |
| 21     | Z         | $\Delta = E = \Gamma = A$ ΑΛΛΑ PID         | PID    |
| 22     | Γ         | $\Delta = E = A$ ΑΛΛΑ PID                  | PID    |
| 23     | A         | $\Delta = E$ ΑΛΛΑ PID                      | PID    |
| 24     | E         | MENEI MONO O Δ                             | PID    |
| 25     | Δ         |  | PID    |

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ: ΧΟ - Χαφ**

| <i><b>ΔΙΕΡΓΑΣΙΑ</b></i> | <b>LR</b>  |
|-------------------------|------------|
| A                       | 24-0=24    |
| B                       | 21-0=21    |
| Γ                       | 23-2=21    |
| Δ                       | 26-3=23    |
| E                       | 25-4=21    |
| Z                       | 22-5=17    |
| <i>ΜΧΟΛΟΚΛΗΡΩΣΗΣ</i>    | 127/6=21,1 |

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ: ΧΔ - ΧΕ**

| <i><b>ΔΙΕΡΓΑΣΙΑ</b></i> | <b>LR</b>  |
|-------------------------|------------|
| A                       | 24-6=18    |
| B                       | 21-4=17    |
| Γ                       | 21-1=20    |
| Δ                       | 26-3=23    |
| E                       | 25-5=20    |
| Z                       | 22-7=15    |
| <i>ΜΧΑΝΑΜΟΝΗΣ</i>       | 113/6=18,8 |

**ΜΕΣΟΣ ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ: Χαρχ - Χαφ**

| <i><b>ΔΙΕΡΓΑΣΙΑ</b></i> | <b>LR</b> |
|-------------------------|-----------|
| A                       | 0-0=0     |
| B                       | 2-0=2     |
| Γ                       | 22-2=20   |
| Δ                       | 14-3=11   |
| E                       | 4-4=0     |
| Z                       | 5-5=0     |
| <i>ΜΧΑΠΟΚΡΙΣΗΣ</i>      | 33/6=5,5  |

**ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ:**

| <i><b>ΔΙΕΡΓΑΣΙΑ</b></i> | <b>LR</b> |
|-------------------------|-----------|
| A                       | 4         |
| B                       | 3         |
| Γ                       | 0         |
| Δ                       | 2         |
| E                       | 4         |
| Z                       | 3         |