ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Αναφορά 1ης Εργαστηριακής Άσκησης

Καμινάρης Κωνσταντίνος, 236064 kaminaris@ceid.upatras.gr

Δεκέμβριος 2023

EPΩTHMA 1: SHELL SCRIPTING

Υλοποιημένα κομμάτια: 1 — 6 Μη υλοποιημένα κομμάτια: —

Κομμάτια που δουλεύουν σωστά: 1-6 Κομμάτια που δε δουλεύουν σωστά: -

Η εφαρμογή υλοποιήθηκε σε BASH script με χρήση awk για την αναζήτηση σε αρχεία.

Το πρόγραμμα αποτελείται από ένα loop που δίνει στο χρήστη τις επιλογές, το οποίο επαναλαμβάνεται μέχρι ο χρήστης να επιλέξει την έξοδο από την εφαρμογή. Αφού ο χρήστης επιλέξει κάποια ενέργεια, η ροή προγράμματος προωθείται μέσω ενός "case ... in" statement στην αντίστοιχη συνάρτηση. Εκεί επιτελείται η εκάστοτε ενέργεια.

Η βασική ιδέα της εφαρμογής είναι πως το αρχείο που θέλει ο χρήστης να διαβάσει/αλλάξει αντιγράφεται σε ένα προσωρινό αρχείο με χρήση της εντολής mktemp. Όλες οι επιλογές του προγράμματος πραγματοποιούνται πάνω σε αυτό το προσωρινό αρχείο, μέχρι να επιλεχθεί η αποθήκευσή του από το χρήστη, οπότε και αντιγράφεται στο "πραγματικό αρχείο".

Ιδιαίτερη προσοχή δόθηκε στην προστασία της εφαρμογής από τυχόν καταρρεύσεις κατά τη διάρκεια της εκτέλεσής της. Παραδείγματα εκτέλεσης των ερωτημάτων:

```
Welcome to Business Hall!

Press <ENTER> to continue...

Select business file - . . . . (1)
Print business data - . . . . (2)
Update business data ield - . . . . . (3)
Print entire file - . . . . . (4)
Save changes - . . . . . . (5)
Exit - . . . . . . . . . . . (6)

Select one of the options above by typing the respective number: 1
Enter the path to your selected business file: businesses.csv

File selected!
Press <ENTER> to continue...
```

Σχήμα 1: Επιλογή αρχείου επιχειρήσεων

Σχήμα 2: Προβολή στοιχείων επιχείρησης

```
Select business file . . . . . (1)
Print business data - . . . (2)
Update business data field . . . (3)
Print entire file . . . . . (4)
Save changes . . . . . . (5)
Exit . . . . . . . . . . (6)

Select one of the options above by typing the respective number: 3
Enter business id: 36363

ID (1)
Name (2)
Adress (3)
City (4)
PostCode (5)
Longitude (6)
Latitude (7)

Select the field you would like to change by entering its respective number (1-7): 2
Enter new field value: McDonalds
Are you sure you want to change the value to McDonalds? (y/n): y
Old data:

ID BusinessName Adress City PostCode Longitude Latitude
36363 1call Direct 116 West Regent Street Glasgow 62 200 -4.258314 55.863552

New data:

ID BusinessName Adress City PostCode Longitude Latitude
36363 McDonalds 116 West Regent Street Glasgow 62 200 -4.258314 55.863552

Update successful!

Press <ENTER> to continue...
```

Σχήμα 3: Αλλαγή στοιχείου επιχείρησης

```
| Company | Comp
```

Σχήμα 4: Προβολή αρχείου

```
| Part |
```

Σχήμα 5: Αποθήκευση αρχείου

```
Select business file - - - - - (1)
Print business data - - - - (2)
Update business data field - - - - (3)
Print entire file - - - - (4)
Save changes - - - - (5)
Exit - - - - (6)

Select one of the options above by typing the respective number: 6

Exit Business Hall? (y/n) :y

Thank you for using Business Hall!
konstantinos: -/osproject/bash-project$
```

Σχήμα 6: Έξοδος

ΕΡΩΤΗΜΑ 2: ΔΙΕΡΓΑΣΙΕΣ

Υλοποιημένα κομμάτια: a, bΜη υλοποιημένα κομμάτια: —

Κομμάτια που δουλεύουν σωστά: a, bΚομμάτια που δε δουλεύουν σωστά: —

Υλοποιήθηκαν τα αρχεία integral mc_shm.c και integral mc_shm_sem.c. Ο αριθμός των διεργασιών ορίζεται στην κλήση των προγραμμάτων με τη μορφή του 1ου argument.

Για τη δημιουργία των διεργασιών παιδιών χρησιμοποιήθηκε η εντολή fork(), ενώ για την κοινή μνήμη η εντολή mmap/munmap. Το seed για τον random number generator τέθηκε σε κάθε διεργασία παιδί σαν συνάρτηση της ώρας και του αριθμού της εργασίας ώστε κάθε seed να είναι διαφορετικό [time(NULL)+childNumber]. Όλες οι διεργασίες παιδιά μοιράζονται το φόρτο υπολογισμού ισότιμα, ενώ η διεργασία γονέας περιμένει την έξοδό τους για να αποκριθεί με το τελικό αποτέλεσμα. Στην περίπτωση της υλοποίησης χωρίς σημαφόρους η κοινή μνήμη αποτελείται από έναν pointer σε μονοδιάστατο πίνακα float με μέγεθος το πλήθος των διεργασιών. Στην περίπτωση με σημαφόρους η κοινή μνήμη αποτελείται από ένα struct με δύο πεδία: το σημαφόρο και τη μοναδική θέση μνήμης που απαιτείται για την αποθήκευση του αποτελέσματος. Η πρόσβαση στην τιμή του αποτελέσματος προστατεύεται από το σημαφόρο.

Παρατήρηση: Όταν οι διεργασίες "έγραφαν" απευθείας στην κοινή μνήμη σε κάθε βήμα του εσωτερικού loop παρατηρήθηκε πτώση στην απόδοση του προγράμματος. Οπότε δημιουργήθηκε μία προσωρινή μεταβλητή partial_sum η οποία ενημερώνει την κρυφή μνήμη μόνο στο τέλος της ζωής της κάθε διεργασίας παιδιού. Ιδιαίτερα σημαντική ήταν η αλλαγή αυτή στην περίπτωση της υλοποίησης με σημαφόρους διότι η κάθε διεργασία παιδί αποκλειόταν για μεγάλο χρόνο έξω από την κρίσιμη περιοχή της με αποτέλεσμα υπέρβαρη καθυστέρηση στην εκτέλεση του προγράμματος.

ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ

Virtual Machine | VMWare Workstation 17 Player

OS | Ubuntu 22.04.3 x64

Host OS | Microsoft Windows 10 Pro

Processor Name | Intel Core i5-4460

Processor Speed 3,2 GHz

Number of Processors 1

Total Number of Cores 4

L2 Cache (per Core) 256 KB

L3 Cache 6 MB Memory 16 GB Καταγράφηκε συστηματική βελτίωση της απόδοσης αναλογικά με το πλήθος των διεργασιών παιδιών μέχρι την τιμή των διαθέσιμων επεξεργαστικών πυρήνων (4). Για μεγαλύτερες τιμές διεργασιών παιδιών δεν παρατηρήθηκε αξιόλογη βελτίωση απόδοσης, όπως ήταν αναμενόμενο. Παράδειγμα εκτέλεσης του κάθε κώδικα με διαφορετικό αριθμό διεργασιών:

```
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm
Result=0.7386400347256546 Error=2.963311e-06 Rel.Error=4.011831e-06 Time=9.518368 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm 2
Result=0.7386416009134991 Error=1.397123e-06 Rel.Error=1.891473e-06 Time=4.665210 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm 3
Result=0.7386451737967854 Error=2.175760e-06 Rel.Error=2.945618e-06 Time=3.136158 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm 4
Result=0.7386438879217652 Error=8.898849e-07 Rel.Error=1.204756e-06 Time=2.440421 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm 10
Result=0.7386536015806203 Error=1.060354e-05 Rel.Error=1.435544e-05 Time=2.521738 seconds
```

Σχήμα 7: Shared memory without semaphore

```
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm_sem
Result=0.7386427352556458 Error=2.627812e-07 Rel.Error=3.557622e-07 Time=9.298959 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm_sem 2
Result=0.7386388656314343 Error=4.132405e-06 Rel.Error=5.594591e-06 Time=4.692712 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm_sem 3
Result=0.7386448054976500 Error=1.807461e-06 Rel.Error=2.447002e-06 Time=3.111875 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm_sem 4
Result=0.7386543432546064 Error=1.134522e-05 Rel.Error=1.535954e-05 Time=2.401062 seconds
konstantinos:-/osproject1/project1_integration$ ./integral_mc_shm_sem 5
Result=0.7386472555510026 Error=4.257514e-06 Rel.Error=5.763967e-06 Time=2.488332 seconds
```

Σχήμα 8: Shared memory with semaphore

ΕΡΩΤΗΜΑ 3: ΣΥΓΧΡΟΝΙΣΜΟΣ ΔΙΕΡΓΑΣΙΩΝ

Αρχικά υπολογίζονται οι σταθερές Χ και Υ από το ΑΜ 1041756 άρα:

$$X = 6$$
 και $Y = 7$

Οι οντότητες που θα υλοποιηθούν είναι οι εξής:

- Συνδαιτημόνας
- Μάγειρας
- Βοηθός 1
- Βοηθός 2

Γνωρίζουμε πως μόνο ένας συνδαιτημόνας μπορεί να σερβίρεται από την κατσαρόλα κάθε φορά και πως όταν ο μάγειρας θέλει να προσθέσει φαγητό θα πρέπει να έχει μόνο αυτός πρόσβαση στην κατσαρόλα, οπότε χρειαζόμαστε ένα δυαδικό σημαφόρο που ονομάζεται pot. Ταυτόχρονα χρειαζόμαστε μια κοινή μεταβλητή food που θα αποθηκεύει πόσες μερίδες απομένουν στην κατσαρόλα. Η πρόσβαση για ανάγνωση ή εγγραφή σε αυτή τη μεταβλητή προστατεύεται από τον δυαδικό σημαφόρο mutex.

Όταν η μεταβλητή food γίνει μηδέν, το φαγητό στην κατσαρόλα έχει αδειάσει και θέλουμε να αφυπνίσουμε τον μάγειρα. Αυτό γίνεται με το δυαδικό σημαφόρο pot_is_empty. Ο μάγειρας αφυπνίζει τους δύο βοηθούς του με τους δυαδικούς σημαφόρους help1 και help2 όσες φορές χρειάζεται, ώστε να έχει όλα τα υλικά που χρειάζεται για να μαγειρέψει. Συγκεκριμένα θα αφυπνίσει μόνο μία φορά τον βοηθό που είναι υπεύθυνος για τα καρότα (βοηθός 2), ενώ περισσότερες φορές τον βοηθό που είναι υπεύθυνος για τα φασόλια (βοηθός 1), γιατί δεν μπορεί να κουβαλήσει περισσότερα από Υ. Για την καταμέτρηση των φασολιών χρησιμοποιείται η κοινή μεταβλητή beans_count η οποία προστατεύεται για ανάγνωση και εγγραφή από το δυαδικό σημαφόρο mutex2.

Ο βοηθός 2 με τη σειρά του φέρνει τα καρότα και ενημερώνει με το δυαδικό σημαφόρο carrots τον μάγειρα πως τα καρότα είναι έτοιμα. Ο βοηθός 1 φέρνει όσα φασόλια μπορεί να κουβαλήσει και ενημερώνει τη μεταβλητή beans_count και μετά ενημερώνει τον μάγειρα πως γύρισε πίσω με το δυαδικό σημαφόρο beans. Όταν ο μάγειρας έχει αρκετά υλικά καταναλώνει τους σημαφόρους beans και carrots και μαγειρεύει. Στη συνέχεια ενημερώνει την κοινή μεταβλητή food να γίνει ίση με X+5. Τέλος αφυπνίζει τους συνδαιτημόνες με το σημαφόρο pot.

Παρατίθεται μια υλοποίηση των παραπάνω σε μορφή ψευδοκώδικα:

```
define X = 6;
define Y = 7;
semaphore pot = 1;
semaphore pot_is_empty = 0;
semaphore help1 = 0;
semaphore help2 = 0;
semaphore beans = 0;
semaphore carrots = 0;
semaphore mutex = 1;
shared int food = X + 5;
semaphore mutex2 = 1;
shared int beans_count = 0;
```

Σχήμα 9: Μεταβλητές

```
Διεργασία-Συνδαιτημόνας

repeat
   down(mutex);
   if (food>0) then
        food—;
        if (food=0) up(pot_is_empty);
        up(mutex);
        down(pot);
        take_food();
        up(pot);
   else up(mutex);

forever;
```

Σχήμα 10: Διεργασία συνδαιτημόνας

Σχήμα 11: Διεργασία μάγειρας

Σχήμα 12: Διεργασίες βοηθοί

ΕΡΩΤΗΜΑ 4: ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΔΙΕΡΓΑΣΙΩΝ

Υλοποιημένα κομμάτια: a, bΜη υλοποιημένα κομμάτια: c

Συμβολισμοί που χρησιμοποιούνται παρακάτω:

 $\begin{array}{ll} {\rm MX}_{{\rm A}\nu} \to & {\rm M\'e}{\rm soc} \ {\rm Cr\'e}{\rm ovoc} \ {\rm An\'amov\'nc} \\ {\rm MX}_{{\rm A}\pi} \to & {\rm M\'e}{\rm soc} \ {\rm Cr\'e}{\rm ovoc} \ {\rm An\'okrishc} \\ {\rm MXO} \to & {\rm M\'e}{\rm soc} \ {\rm Cr\'e}{\rm ovoc} \ {\rm Olokl\'nr} \\ \Theta {\rm A} \to & {\rm Πλ\'n}\theta {\rm oc} \ \Theta {\rm ematik\'nv} \ {\rm Allary\'mv} \end{array}$

FIRST COME FIRST SERVED (FCFS)

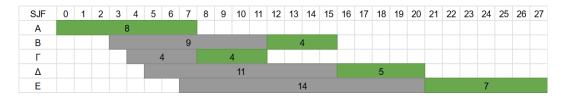


Σχήμα 13: FCFS Gantt Chart

$$\begin{split} \mathrm{MX_{A\nu}} &= \frac{0+5+8+11+14}{5} = 7.6 \\ \mathrm{MX_{A\pi}} &= \frac{0+5+8+11+14}{5} = 7.6 \\ \mathrm{MXO} &= \frac{8+9+12+16+21}{5} = 13.2 \\ \Theta\mathrm{A} &= 1+1+1+1+1=5 \end{split}$$

SHORTEST JOB FIRST (SJF)

Εδώ ενεργοποιείται η προτεραιότητα των διεργασιών τη χρονική στιμγή 8 καθώς οι διεργασίες Β και Γ έχουν διάρκεια εκτέλεσης 4 μονάδες η κάθε μία αλλά η Γ έχει μεγαλύτερη προτεραιότητα.

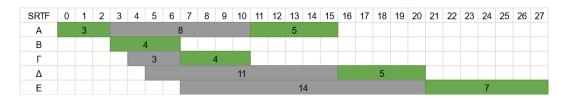


Σχήμα 14: SJF Gantt Chart

$$\begin{split} \mathrm{MX}_{\mathrm{A}\nu} &= \frac{0+9+4+11+14}{5} = 7.6\\ \mathrm{MX}_{\mathrm{A}\pi} &= \frac{0+9+4+11+14}{5} = 7.6\\ \mathrm{MXO} &= \frac{8+13+8+16+21}{5} = 13.2\\ \Theta\mathrm{A} &= 1+1+1+1+1=5 \end{split}$$

SHORTEST REMAINING TIME FIRST (SRTF)

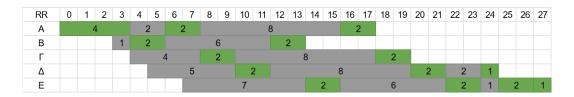
Εδώ ενεργοποιείται η προτεραιότητα των διεργασιών τη χρονική στιμγή 11 καθώς οι διεργασίες Α και Δ έχουν υπόλοιπη διάρκεια εκτέλεσης 5 μονάδες η κάθε μία αλλά η Α έχει μεγαλύτερη προτεραιότητα.



Σχήμα 15: SRTF Gantt Chart

$$\begin{split} \mathrm{MX}_{\mathrm{A}\nu} &= \frac{8+0+3+11+14}{5} = 7.2 \\ \mathrm{MX}_{\mathrm{A}\pi} &= \frac{0+0+3+11+14}{5} = 5.6 \\ \mathrm{MXO} &= \frac{16+4+7+16+21}{5} = 12.8 \\ \mathrm{\Theta A} &= 2+1+1+1+1 = 6 \end{split}$$

ROUND-ROBIN (RR)



Σχήμα 16: RR Gantt Chart

$$\begin{split} \mathrm{MX_{A\nu}} &= \frac{10+7+12+15+14}{5} = 11.6 \\ \mathrm{MX_{A\pi}} &= \frac{0+1+4+5+7}{5} = 3.4 \\ \mathrm{MXO} &= \frac{18+11+16+20+21}{5} = 17.2 \\ \Theta\mathrm{A} &= 3+2+2+3+3 = 13 \end{split}$$