Operating Systems - Project 2

Προκοπίου Ιωάννης	1059554
Σίνα Ιωάννης	1059610
Κονταρίνης Απόστολος	1059565
Κουνέλης Αγησίλαος	1059637

Μέρος Α

Ερώτημα Α:

i. Σε κατάσταση sleeping 10 δευτερόλεπτα μετά την έναρξή του προγράμματος υπάρχουν 4 διεργασίες. Η γονική διεργασία εκτελεί την εντολή fork (pid1 = fork();). Έτσι δημιουργεί μία διεργασία παιδί. Έπειτα και οι δύο διεργασίες εισέρχονται στο else και η κάθε μία εκτελεί την εντολή fork (pid2 = fork();). Επομένως συνολικά έχουμε 4 διεργασίες (τη διεργασία πατέρα, τα δύο της παιδιά και το ένα της εγγόνι).

ii.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
	int pid1;
	int pid2;
	pid1 = fork();
	if (pid1 < 0)
		printf("Could not create any child\n");
	else {
		pid2 = fork();
		if (pid2 < 0)printf("Could not create any child\n");
		else if ((pid1 < 0) && (pid2 < 0))
			kill(pid1,9);
	}

	printf("My Parent is: %d and i am : %d\n",getppid(),getpid());
	sleep(20);
	return (0);
}
```

Ερώτημα Β:

Κάθε διεργασία προσπαθεί μέσω της συνάρτησης insert_key να μπει στο heap και για το λόγο αυτό χρησιμοποιούμε τον δυαδικό σημαφόρο mutex, ο οποίος επιτρέπει σε μία διεργασία τη φορά να χρησιμοποιεί την συνάρτηση αυτή, που αποτελεί την κρίσιμη περιοχή. Η χρήση του mutex εμποδίζει την ταυτόχρονη εισαγωγή δύο διεργασιών στο heap με αμφίβολα αποτελέσματα. Η τιμές των μεταβλητών priority καθορίζουν τη σειρά των διεργασιών στο heap.
Αρχείο: erwthmaB.c

Ερώτημα Γ:

Δημιουργούμε μία διεργασία παιδί που τη χρησιμοποιούμε ως reader. Επίσης, χρησιμοποιούμε τη διεργασία πατέρα ως writer. Μπορούμε να έχουμε πολλούς αναγνώστες γιατί χρησιμοποιούμε έναν δυαδικό σημαφόρο και έναν shared integer. Αυτό που συμβαίνει είναι ότι όταν μπαίνει ο πρώτος αναγνώστης και κάνει down τον σημαφόρο της βάσης μπορούν να μπουν όσοι άλλοι αναγνώστες θέλουν. Όταν όλοι αυτοί τελειώσουν μπορεί να γράψει στην βάση η διεργασία εγγραφή.
Αρχείο: erwthmaC.c

Ερώτημα Δ:

(1)

- Λύση 1 – counter semaphores

Για να επιτύχουμε το συγχρονισμό των 5 διεργασιών παρατηρούμε 3 “επίπεδα” διεργασιών {P1, P2}, {P3}, {P4, P5} και έτσι αρχικοποιούμε δύο counter σημαφόρους τους sem1 και sem2. Η διαδικασία είναι:

- Αρχικοποίηση sem1 και sem2 με 0
- Μετά την εκτέλεση των P1 και P2 γίνεται από μία φορά up για τον sem1
- Πριν την εκτέλεση της P3 γίνεται δύο φορές down ο sem1
- Μετά την εκτέλεση της P3 γίνεται δύο φορές up ο sem2
- Πριν την εκτέλεση των P4 και P5 γίνεται από μία φορά down ο sem2

Αρχείο: erwthmaD1v1.c

- Λύση 2 – binary semaphores

Για να επιτύχουμε το συγχρονισμό των 5 διεργασιών παρατηρούμε ότι μπορούμε να αντικαταστήσουμε κάθε ακμή με ένα binary σημαφόρο. Η διαδικασία είναι:

- Αρχικοποίηση όλων με 0
- Μετά την εκτέλεση των P1 και P2 γίνεται up ο sem13 και ο sem23 αντίστοιχα
- Πριν την εκτέλεση της P3 γίνεται down των sem13 και sem23 αντίστοιχα
- Μετά την εκτέλεση της P3 γίνεται up ο sem34 και ο sem35
- Πριν την εκτέλεση των P4 και P5 γίνεται down των sem34 και sem35 αντίστοιχα

Αρχείο: erwthmaD1v2.c

(2)
Ομοίως με 4 counter ή 6 binary σημαφόρους.
Αρχεία: erwthmaD2v1.c και erwthmaD2v2.c

Μέρος Β

Ερώτημα Α:

α)

Αριθμός Λογικής Σελίδας: 18bits	Διεύθυνση μέσα στη σελίδα: 14bits

Θεωρούμε ότι κάθε θέση μνήμης αποτελείται από 1 byte.
Μέγεθος Διεργασίας: 39500(16) bytes = 234752 bytes.
Μέγεθος Πλαισίου Σελίδας: 2^14 bytes (έχει 2^14 γραμμές και κάθε γραμμή είναι του 1 byte)
#Πλαισίων σελίδων = μέγεθος διεργασίας/μέγεθος πλαισίου=⌈234752/2^14)=(14.328125)=15

Εσωτερική κλασματοποίηση είναι το μέγεθος της μνήμης το οποίο ξοδεύεται μέσα σε κάποιες περιοχές που έχουν δεσμευθεί.
Άρα η εσωτερική κλασματοποίηση είναι 15*2^14 - 234752 = 11008 bytes
β)i)
00031958 = 0000 0000 0000 0011 00**01 1001 0101 1000**

Μετά τον διαχωρισμό του αριθμού σελίδας και της **μετατόπισης** έχουμε:
Αριθμός σελίδας: 0000 0000 0000 0000 1100 = 12 (dec)
Μετατόπιση: 01 1001 0101 1000

Η 12η σελίδα έχει φορτωθεί στη μνήμη.
Τα δεδομένα που περιέχει η 12η θέση του πίνακα σελίδων περιέχει τον αριθμό 225 (dec).
Επομένως ο αριθμός πλαισίου σελίδας είναι 225 = 1110 0001
Άρα η φυσική διεύθυνση είναι: 0011 1000 01**01 1001 0101 1000** = 385958 (hex)

ii)
0001E800 = 0000 0000 0000 0001 11**10 1000 0000 0000**

Μετά τον διαχωρισμό του αριθμού σελίδας και της **μετατόπισης** έχουμε:
Αριθμός σελίδας: 0000 0000 0000 0000 0111 = 7 (dec)
Μετατόπιση: 10 1000 0000 0000

Η 7η σελίδα ΔΕΝ έχει φορτωθεί στη μνήμη κατά το τρέχον χρονικό διάστημα.
Επομένως δεν υπάρχει ο ζητούμενος αριθμός πλαισίου σελίδας στην 7η σελίδα και δεν μπορεί να βρεθεί η αντίστοιχη φυσική διεύθυνση. Δηλαδή έχουμε page fault και το λειτουργικό σύστημα αναλαμβάνει να φέρει τη ζητούμενη σελίδα και να ενημερώσει τον πίνακα σελίδων. Αυτή η διαδικασία λέγεται προσκόμιση σελίδας κατ' απαίτηση (on demand paging).

Ερώτημα Β:

Αριθμός Λογικού Τμήματος: 8bits	Διεύθυνση μέσα στο τμήμα: 24bits

Φυσική Διεύθυνση Τμήματος: -bits	Διεύθυνση μέσα στο τμήμα: 24bits

Μέγεθος τμήματος = 16 MBytes = 2^(4+20) bytes = 2^24 bytes.
Θεωρούμε ότι κάθε θέση μνήμης αποτελείται από 1 byte.
Άρα 24 bits για διεύθυνση μέσα στο τμήμα.
Άρα ο αριθμός λογικού τμήματος είναι 32-24 = 8bits.

α) Θεωρούμε μία διεργασία που καταλαμβάνει το μέγιστο αριθμό λογικών τμημάτων που υπάρχει διαθέσιμος. Αφού ο αριθμός λογικού τμήματος αποτελείται από 8bits το πλήθος των λογικών τμημάτων θα είναι 2^8 = 256(dec).

β)i)
0B00042A = 0000 1011 **0000 0000 0000 0100 0010 1010**
Μετά τον διαχωρισμό του αριθμού τμήματος και της **μετατόπισης** έχουμε:
Αριθμός Λογικού Τμήματος: 0000 1011 = 11 (dec)
Μετατόπιση: 0000 0000 0000 0100 0010 1010 = 1066 (dec)

Το μήκος τμήματος της 11ης θέσης του πίνακα τμημάτων είναι μεγαλύτερο της μετατόπισης, επομένως η προσπέλαση είναι έγκυρη.
Η 11η θέση του πίνακα τμημάτων έχει διεύθυνση βάσης 9050 (dec).
Τελικά η φυσική διεύθυνση είναι: 9050 + 1066 = 10116(dec) = 0010 0111 1000 0100 = 2784(hex)

ii)
02000B6D = 0000 0010 **0000 0000 0000 1011 0110 1101**
Μετά τον διαχωρισμό του αριθμού τμήματος και της **μετατόπισης** έχουμε:
Αριθμός Λογικού Τμήματος: 0000 0010 = 2 (dec)
Μετατόπιση: 0000 0000 0000 1011 0110 1101 = 2925 (dec)

Το μήκος τμήματος της 2ης θέσης του πίνακα τμημάτων είναι **μικρότερο** της μετατόπισης, επομένως η προσπέλαση **ΔΕΝ** είναι έγκυρη. Άρα έχουμε σφάλμα τμήματος και δεν μπορεί να σχηματιστεί η επιθυμητή φυσική διεύθυνση.

Ερώτημα Γ:
ΣΕΛΙΔΟΠΟΙΗΜΈΝΗ ΤΜΗΜΑΤΟΠΟΙΗΣΗ

α)
Παραμένει s = 8 bits(#λογικών τμημάτων)
Είχαμε d = 24 bits
Άρα αφού μέγεθος σελίδας = 512 bytes = 2^9 bytes
θα έχουμε: d = 9 bits, p = 24 - 9 = 15 bits

Αριθμός Τμήματος: 8 bits	Αριθμός Σελίδας: 15 bits	Μετατόπιση: 9 bits

β)
Θεωρούμε μία διεργασία που καταλαμβάνει το μέγιστο αριθμό λογικών σελίδων που υπάρχει διαθέσιμος. Αφού ο αριθμός λογικών σελίδων αποτελείται από 15 bits και το πλήθος των λογικών τμημάτων ειναι 8 θα έχουμε 2^8 λογικά τμήματα * 2^15 λογικές σελίδες.

2^8 * 2^15 = 2^23 = 8388608(dec) σελίδες

γ)i)
010004CF = 00000001 000000000000010 011001111

- Αριθμός Τμήματος = 00000001 = 1
  Άρα πηγαίνουμε στον πίνακα σελίδων τμήματος 1
- Αριθμός Σελίδας = 000000000000010 = 2
  Άρα πηγαίνουμε στη σελίδα 2
  Παρατηρούμε ότι η σελίδα είναι φορτωμένη στη μνήμη και το περιεχόμενο της 2ης θέση είναι: 0B0B = 0000 1011 0000 1011
- Μετατόπιση = 011001111

Άρα η φυσική διεύθυνση είναι ο συνδυασμός του αριθμού πλαισίου και της μετατόπισης μέσα σε αυτό:

0001 0110 0001 0110 1100 1111 = 1616CF

ii)
- 010009FF = 0000 0001 0000 0000 0000 1001 1111 1111
  - Αριθμός Τμήματος = 00000001 = 1
  - Αριθμός Σελίδας = 000000000000100 = 4
  - Μετατόπιση = 1 1111 1111

→ Επειδή η δοσμένη λογική διεύθυνση προκαλεί page fault ο Αριθμός πλαισίου του Αριθμού Σελίδας 4 στο τμήμα 1 θα είναι -

- 000003F0 = 0000 0000 0000 0000 0000 0011 1111 0000
  - Αριθμός Τμήματος = 00000000 = 0
  - Αριθμός Σελίδας = 000000000000001 = 1
  - Μετατόπιση = 1 1111 0000

→ Αν από τη φυσική διεύθυνση κρατήσουμε μόνο τον Αριθμό Πλαισίου και αγνοήσουμε τη μετατόπιση (9 bits) έχουμε:

E0E1F0 = **1110 0000 1110 000**1 1111 0000

Άρα ο αριθμός πλαισίου του αριθμού Σελίδας 1 στο τμήμα 0 θα είναι 0111 0000 0111 0000 = 7070 (hex).

Ερώτηση Δ:

Στρατηγική αντικατάστασης: **LRU**

	+3	+5	+8	+1	+8	+7	+5	+1	+8	+2	+4	+2	+7	+3	+6	+4	+7	+5	+3	+7
-	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
0	3	3	3	3	3	5	1	8	7	5	1	1	8	4	2	7	3	6	4	4
1	-	5	5	5	5	1	8	7	5	1	8	8	4	2	7	3	6	4	7	5
2	-	-	8	8	1	8	7	5	1	8	2	4	2	7	3	6	4	7	5	3
3	-	-	-	1	8	7	5	1	8	2	4	2	7	3	6	4	7	5	3	7

Στο τρίτο πλαίσιο έχουμε πάντα την σελίδα που χρησιμοποιήθηκε πιο πρόσφατα. Η υλοποίηση αυτή είναι χρονικά ακριβή επειδή η εναλλαγή των πληροφοριών μεταξύ των πλαισίων γίνεται συχνά. Εναλλακτικά προτείνεται να διατήρηση χρονικής πληροφορίας (time stamp) για κάθε σελίδα.