ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

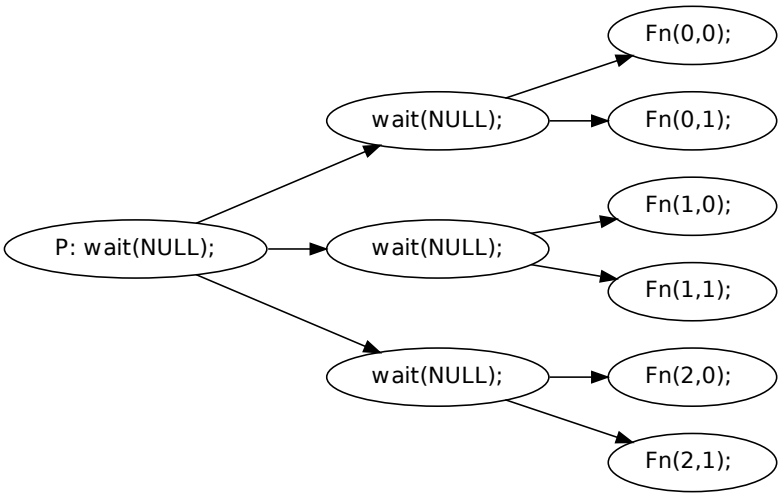
Λειτουργικά Συστήματα

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2009-2010

Φεβρουάριος 2010 – Λύσεις Θεμάτων

Θέμα 1

α.



β.

```
ret = fork();
if (ret == 0){
  B();
  assert(0); // should never reach here
}

ret = fork();
if (ret == 0){
  ret = fork();
  if (ret == 0){
    D();
    assert(0); // should never reach here
  }
  C();
  assert(0); //should never reach here
}

A();
assert(0); // should never reach here
```

Θέμα 2

α.

```
SemA = Semaphore(0);
SemB = Semaphore(0);

T0():                          T1():
    A0();                          A1();
    signal(SemB);                  signal(SemA);
    wait(SemA);                    wait(SemB);
    B0();                          B1();
```

β.

i.

Έστω η παρακάτω εκτέλεση, για δύο νήματα:

T_0	T_1	
`A0();`		
`wait(lock);`		
`count++`		$count = 1$
`signal(lock);`		
`wait(barrier);`		$count \neq 2$
	`A1();`	
	`wait(lock);`	
	`count++`	$count = 2$
	`signal(lock);`	
	`signal(barrier);`	$count == 2$
`B0();`		
	`wait(barrier);`	

Το νήμα T_1 δεν θα επιστρέψει από το `wait(barrier)`.

ii.

Ναι. Αν κάθε `count = count + 1` πραγματοποιηθεί για **όλα** τα νήματα, **πριν** κάθε έλεγχο `count == N`.

Για παράδειγμα για δύο νήματα:

T_0	T_1	
`A0();`		
`wait(lock);`		
`count++`		$count = 1$
`signal(lock);`		
	`A1();`	
	`wait(lock);`	
	`count++`	$count = 2$
	`signal(lock);`	
`signal(barrier);`		$count == 2$
	`signal(barrier);`	$count == 2$
`wait(barrier);`		
	`wait(barrier);`	

iii.

Προστατεύει τη μεταβλητή `count` από ταυτόχρονη πρόσβαση.

iv.

```
wait(lock);
count++;
signal(lock);

if (count == N);
    signal(barrier);

wait(barrier);
signal(barrier);
```

1 Θέμα 3

1.1 α.

t	Ουρά ΚΜΕ	ΚΜΕ	Ουρά Ε/Ε	Ε/Ε
0	$P_1/10$	P_1		-
2	$P_1/8$,$P_2/6$	P_2	-	-
4	$P_1/8$,$P_2/4$,$P3/1$	P_3	-	-
5	$P_1/8$,$P_2/4$	P_2	$P_3/1$	P_3
6	$P_1/8$,$P_2/3$,$P_3/2$	P_3	-	-
8	$P_1/8$,$P_2/3$	P_2	$P_3/3$	P_3
11	$P_1/8$	P_1	$P_2/5$	P_2
16	$P_1/3$,$P_2/2$	P_2	-	-
18	$P_1/3$	P_1	$P_2/2$	P_2
20	$P_1/1$	P_1	-	-
21	-	-	$P_1/10$	P_1
31	-	-	-	-

1.2 β.

t	Ουρά ΚΜΕ	ΚΜΕ #1	ΚΜΕ #2	Ουρά Ε/Ε	Ε/Ε
0	$P_1/10$	P_1	-		-
2	$P_1/8$,$P_2/6$	P_2	P_1	-	-
4	$P_1/6$,$P_2/4$,$P_3/1$	P_3	P_2	-	-
5	$P_1/6$,$P_2/3$	P_2	P_1	$P_3/1$	P_3
6	$P_1/5$,$P_2/2$,$P_3/2$	P_2	P_3	-	-
8	$P_1/5$	P_1	-	$P_3/3$,$P_2/5$	P_2
13	$P_2/2$	P_2	-	$P_1/10$,$P_3/3$	P_3
15	-	-	-	$P_2/2$,$P_1/10$,$P_3/1$	P_3
16	-	-	-	$P_2/2$,$P_1/10$	P_1
18	-	-	-	$P_2/2$,$P_1/8$	P_1
26	-	-	-	$P_2/2$	P_2
28	-	-	-	-	-

γ.

i.

$$u_0 = \frac{21}{31} = .68$$

ii.

$$u_0 = \frac{15}{28} = .54$$

$$u_1 = \frac{6}{28} = .21$$

iii.

Ο συνολικός χρόνος δεν είναι ο μισός, διότι οι διεργασίες σειριοποιούνται στη μονάδα Ε/Ε.

Θέμα 4

α.

- Βέλτιστή

Αναφορά	Διαθέσιμα πλάισια μνήμης	σφάλμα
1	1	ΝΑΙ
3	3,1	ΝΑΙ
4	4,3,1	ΝΑΙ
3	4,3,1	
6	6,3,1	ΝΑΙ
1		
3		
4	4,3,1	ΝΑΙ
3		

- LRU

Αναφορά	Διαθέσιμα πλάισια μνήμης	σφάλμα
1	1	NAI
3	3,1	NAI
4	4,3,1	NAI
3	3,4,1	
6	6,3,4	NAI
1	1,6,3	NAI
3	3,1,6	
4	4,3,1	NAI
3	3,4,1	
		6

- FIFO

Αναφορά	Διαθέσιμα πλάισια μνήμης	σφάλμα
1	1	NAI
3	3,1	NAI
4	4,3,1	NAI
3	4,3,1	
6	6,4,3	NAI
1	1,6,4	NAI
3	3,1,6	NAI
4	4,3,1	NAI
3	4,3,1	
		7

β.

Όσο περιμένει τα δεδομένα από το δίσκο είναι **Σε αναμονή**, γιατί δεν μπορεί να χρησιμοποιήσει την ΚΜΕ. Όταν έρθει η σελίδα στη μνήμη, μπορεί πλέον να χρησιμοποιήση την ΚΜΕ και θα γίνει **Έτοιμη**.

γ.

Η διεργασία-παιδί που δημιουργείται από τη `fork()` αποτελεί αντίγραφο της διεργασίας-πατέρας. Ωστόσο, τα δεδομένα της διεργασίας δεν πρόκειται να χρησιμοποιηθούν διότι η `exec()` τα ακυρώνει (γράφει νέα). Η τεχνική COW αποτρέπει την αντιγραφή των δεδομένων μέχρι να γίνει κάποια εγγραφή και κατα συνέπεια αποφεύγει την άχρηστη αντιγραφή.