ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Λειτουργικά Συστήματα

Φροντιστηριακή ενότητα # 4 : Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

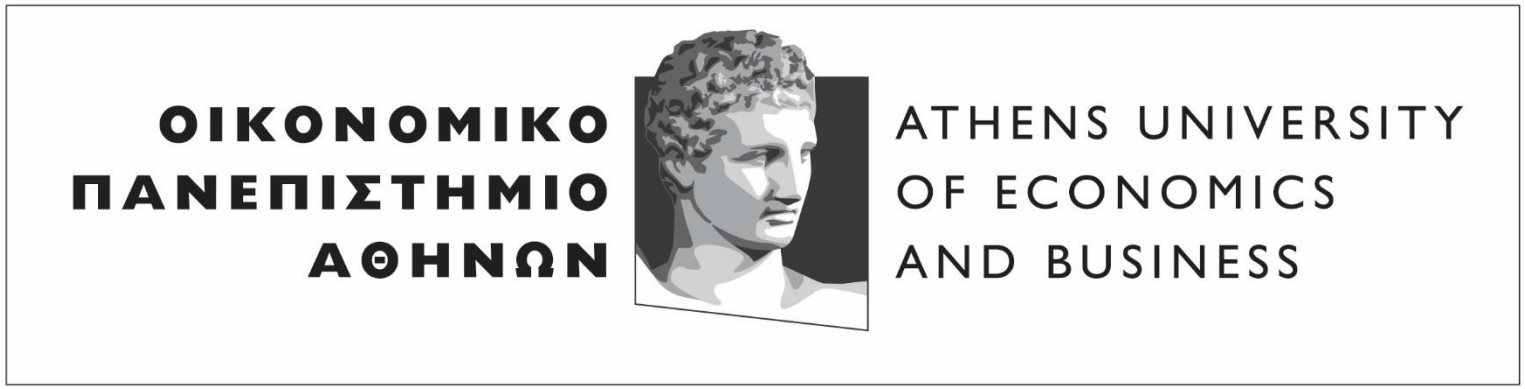
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σκοποί ενότητας

- Εισαγωγή στον πολυνηματικό προγραμματισμό
- Εισαγωγή στα POSIX Threads και το συγχρονισμό τους
- Κατανόηση του προβλήματος παραγωγού – καταναλωτή
- Αποφυγή κοινών σφαλμάτων

Περιεχόμενα

- Νήματα POSIX
- Συγχρονισμός νημάτων
- Παράδειγμα: Παράλληλες παραγγελίες
- Κοινά σφάλματα
- Ψευδοτυχαίοι αριθμοί
- Βιβλιογραφία



Νήματα POSIX

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Δημιουργία Νήματος

- int pthread_create(pthread_t *thread, pthread_attr_t *attr, void *(*start_routine)(void *), void *arg);
  - thread: μοναδικό αναγνωριστικό για το νέο νήμα.
  - attr: αντικείμενο thread_attr_t ή NULL για τις default τιμές.
  - start_routine: η C ρουτίνα που θα εκτελέσει το νέο νήμα.
  - arg: παράμετρος που θα περάσει στην start_routine.

Τερματισμός Νήματος

- void pthread_exit(void *value_ptr);
  - Καλείται από το ίδιο το νήμα (self-destruction)
  - Εναλλακτικά: Τερματίζει κανονικά η ρουτίνα start_routine (“return NULL;”)
  - ΠΡΟΣΟΧΗ στο exit(): τερματίζει όλη τη διεργασία!
- int pthread_cancel(pthread_t my_thread);
  - Καλείται από το νήμα της main() και σκοτώνει το my_thread
  - ΠΡΟΣΟΧΗ στα memory leaks!
- int pthread_join(pthread_t my_thread, void **value_ptr);
  - Καλείται από το νήμα της main(), το οποίο παραμένει αδρανές (suspended) μέχρι να τερματίσει το νήμα my_thread

Attributes νήματος (1 από 3)

- Τα attributes είναι ένας τρόπος να παραμετροποιηθεί η συμπεριφορά των νημάτων.
- Είναι αυτόνομα αντικείμενα που μπορούν να συνδεθούν με πολλά νήματα.
- Η σύνδεση μπορεί να γίνει μόνο κατά την δημιουργία ενός νήματος (pthread_create())
- Τα default attributes είναι συνήθως η καλύτερη επιλογή.

Attributes νήματος (2 από 3)

- Detach state
  - int pthread_attr_setdetachstate(pthread_attr_t *tattr,int detachstate);
    - Τα nondetached νήματα δεν αποδεσμεύουν τους πόρους που κατέχουν αν δεν κληθεί η pthread_join() (προσοχή: memory leak!)
- Scope
  - int pthread_attr_setscope(pthread_attr_t *tattr,int scope);
    - Ορίζει το περιβάλλον ανταγωνισμού (system or process) του νήματος για δέσμευση πόρων πχ CPU.

Attributes νήματος (3 από 3)

- Scheduling Policy
  - int pthread_attr_setschedpolicy(pthread_attr_t *tattr, int policy);
    - Επιλογές: FIFO, RR or OTHER, δεν υποστηρίζεται σε όλα τα ΛΣ
  - int pthread_attr_setinheritsched(pthread_attr_t *tattr, int inherit);
    - Τα καινούργια νήματα (δεν) κληρονομούν την πολιτική του πατέρα
  - int pthread_attr_setschedparam(pthread_attr_t *tattr, const struct sched_param *param);
    - Μόνο η sched_priority υποστηρίζεται

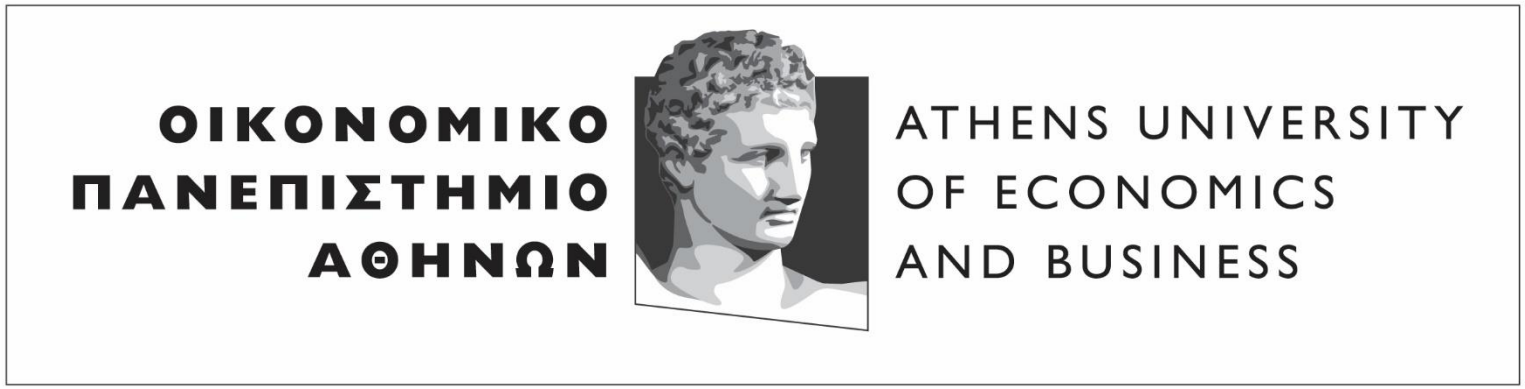
Start Routine

- void * start_routine(void * arg);
  - Εκτελείται από το νήμα αμέσως μετά την εκτέλεση της pthread_create()
  - Επιστρέφει void δείκτη, ανάλογο με την τιμή που επιστρέφει η pthread_exit()
    - Για να δούμε την τιμή, πρέπει να κάνουμε join με το νήμα
  - Δέχεται δείκτη σε void αντικείμενο, το οποίο θα γίνει cast
  - Σε περίπτωση που ένα νήμα χρειάζεται πολλές παραμέτρους πρέπει να δημιουργηθεί δομή (struct)

Βασικό παράδειγμα Pthreads

```
1.  #include <stdio.h>
2.  #include <pthread.h>
3.  #include <unistd.h>
4.  int main()  {
5.     pthread_t t1, t2;
6.     void *f2(), *f1();
7.     int i1,i2;
8.     i1 = 1;
9.     i2 = 2;
10.    pthread_create(&t1, NULL, f1, &i1);
11.    pthread_create(&t2, NULL, f2, &i2);
12.    pthread_join(t1, NULL);
13.    pthread_join(t2, NULL);
14. }
```

```
14.   void *f1(void *x){
15.     int i = *(int*)x;
16.     usleep(1000);
17.     printf("f1: %d\n",i);
18.     pthread_exit(NULL);
19.   }
20.   void *f2(void *x){
21.     int i = *(int*)x;
22.     usleep(1000);
23.     printf("f2: %d\n",i);
24.     return NULL;
25.   }
```



Συγχρονισμός νημάτων

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Thread-safe

- Διαχείριση των διαμοιραζόμενων πόρων με τρόπο που να αποκλείονται ασυνέπειες στην κατάσταση της μνήμης όταν πολλαπλά νήματα λειτουργούν ταυτόχρονα.
- Περίπλοκο ζήτημα συνέπειας που συναντάται στις ΒΔ.
- Δύσκολο πρόβλημα, με ξεκάθαρη λύση μόνο σε αυστηρά καθορισμένα σενάρια χρήσης.
- Συνήθως δίνεται περισσότερη προσοχή στην τροποποίηση ενός πόρου και όχι στην ανάγνωση. Τι γίνεται όμως όταν η ανάγνωση μεταβάλει την κατάσταση του πόρου (βλ. μνήμη cache);

Αμοιβαίος Αποκλεισμός

- Ένα νήμα ενημερώνει τα υπόλοιπα ότι θέλει/έχει αποκλειστική πρόσβαση σε πόρο!
  - Χρησιμοποιούμε αντικείμενο τύπου pthread_mutex_t, έστω lock.
  - Το lock είναι εννοιολογικά συνδεδεμένο με τον thread-safe πόρο.
  - Κάθε νήμα πριν προσπελάσει τον πόρο καλεί την pthread_mutex_lock(lock) και τον δεσμεύει.
  - Όταν δεν τον χρειάζεται καλεί την pthread_mutex_unlock(lock) και τον αποδεσμεύει.
  - Αν εν τω μεταξύ άλλο νήμα καλέσει την pthread_mutex_lock(lock) θα περιμένει μέχρι το πρώτο νήμα να καλέσει την unlock.
  - * pthread_mutex_trylock() – προσπαθεί να κλειδώσει νήμα ή επιστρέφει κωδικό σφάλματος αν αποτύχει το κλείδωμα.

Βασικό παράδειγμα Mutex

```
1.  #include <stdio.h>
2.  #include <pthread.h>
3.  int counter = 0;
4.  pthread_mutex_t lock;
5.  void *f(void *x);
6.  Int main()  {
7.     pthread_t t1, t2;
8.     int i1 = 1;   int i2 = 2;
9.     pthread_mutex_init(&lock, NULL);
10.    pthread_create(&t1, NULL, &f, &i1);
11.    pthread_create(&t2, NULL, &f, &i2);
12.    pthread_join(t1, NULL);
13.    pthread_join(t2, NULL);
14.    pthread_mutex_destroy(&lock);
15. }
```

```
15. void *f(void *x){
16.   int my_id = *(int*)x;
17.   pthread_mutex_lock(&lock);
18.   if (counter==0){
19.         counter+=my_id;
20.         printf("New counter %d \n",
                                    counter);
21.   }
22.   pthread_mutex_unlock(&lock);
23.   return NULL;
24. }
```

Εξασφαλίζουμε ότι μόνο ένα νήμα θα αυξήσει τον counter σε κάθε στιγμή!

Condition Variables (1 από 2)

- Βασική ιδέα: το thread A παραμένει αδρανές όσο κάποια συνθήκη είναι αναληθής (false)
- Κάποια στιγμή ένα άλλο thread θα αλλάξει την συνθήκη και θα στείλει σήμα στο αδρανές thread A ότι μπορεί να συνεχίσει
- Συνεπώς, έχουμε άμεση επικοινωνία μεταξύ νημάτων
- Και εδώ χρησιμοποιούμε ένα αντικείμενο pthread_mutex_t, έστω lock
- Επιπλέον, χρησιμοποιούμε ένα αντικείμενο τύπου pthread_cond_t, έστω cond

Condition Variables (2 από 2)

- Τα νήματα κάνουν lock & unlock το mutex που σχετίζεται με τον πόρο.
- Το 1ο νήμα, που θα περιμένει, καλεί την pthread_cond_wait(cond, lock) μέσα σε loop, προς αποφυγήν spurious wakeups.
- Η pthread_cond_wait() αποδεσμεύει το mutex και έτσι το 2ο νήμα αποκτά πρόσβαση.
- Το 2ο νήμα αλλάζει την συνθήκη, καλεί την pthread_cond_signal(cond) και απελευθερώνει το lock και αργότερα και το mutex.
- Το 1ο νήμα συνεχίζει. Χρησιμοποιεί τον πόρο και απελευθερώνει το mutex.
- * pthread_cond_broadcast() αντί του pthread_cond_signal() όταν θέλω να ξυπνήσω πολλά νήματα!
  - Για παράδειγμα, όταν περιμένουν για διαφορετικό πλήθος πόρων.
  - Άρα πρέπει όλα να δοκιμάσουν αν ικανοποιείται η συνθήκη τους.

Παράδειγμα Condition Variable

```
1.  #include <stdio.h>
2.  #include <pthread.h>
3.  int counter = 0;
4.  pthread_mutex_t lock;
5.  pthread_cond_t  cond  =
    PTHREAD_COND_INITIALIZER;
6.  void *f1(void *), *f2(void *);
7.  main()  {
8.      pthread_t t1, t2;
9.      pthread_mutex_init(&lock, NULL);
10.     pthread_create(&t1, NULL, &f1, NULL);
11.     pthread_create(&t2, NULL, &f2, NULL);
12.     pthread_join(t1, NULL);
13.     pthread_join(t2, NULL);
14.     pthread_mutex_destroy(&lock);
15.     pthread_cond_destroy(&cond);
16. }
```

```
16. void *f1(void* unused){
14.   pthread_mutex_lock(&lock);
15.   while (counter==0)
16.       pthread_cond_wait(&cond, &lock);
17.   counter = 5000 / counter;
18.   printf("New counter: %d\n", counter);
19.   pthread_mutex_unlock(&lock);
20.   return NULL; } //end f1()

21. void *f2(void* unused){
22.   pthread_mutex_lock(&lock);
23.   counter = 1000;
24.   printf("First update: %d\n",counter);
25.   pthread_cond_signal(&cond);
26.   pthread_mutex_unlock(&lock);
27.   return NULL; } //end f2()
```

"while" αντί για "if" ώστε να αντιμετωπί-σουμε spurious wakeups!

Αδιέξοδο (1/2)

```
1.  void *function1()
2.  {
3.  pthread_mutex_lock(&lock1);          // step 1
4.  pthread_mutex_lock(&lock2);          // step 3
    pthread_mutex_unlock(&lock2);
5.  pthread_mutex_unlock(&lock1);
6.  }
7.
8.  void *function2()
9.  {
10. pthread_mutex_lock(&lock2);          // step 2
11. pthread_mutex_lock(&lock1);
12. pthread_mutex_unlock(&lock1);
13. pthread_mutex_unlock(&lock2);
14. }
15.
16. main()
17. {
18. pthread_create(&thread1, NULL, function1,
    NULL);
19. pthread_create(&thread2, NULL, function2,
    NULL);
20. }
```

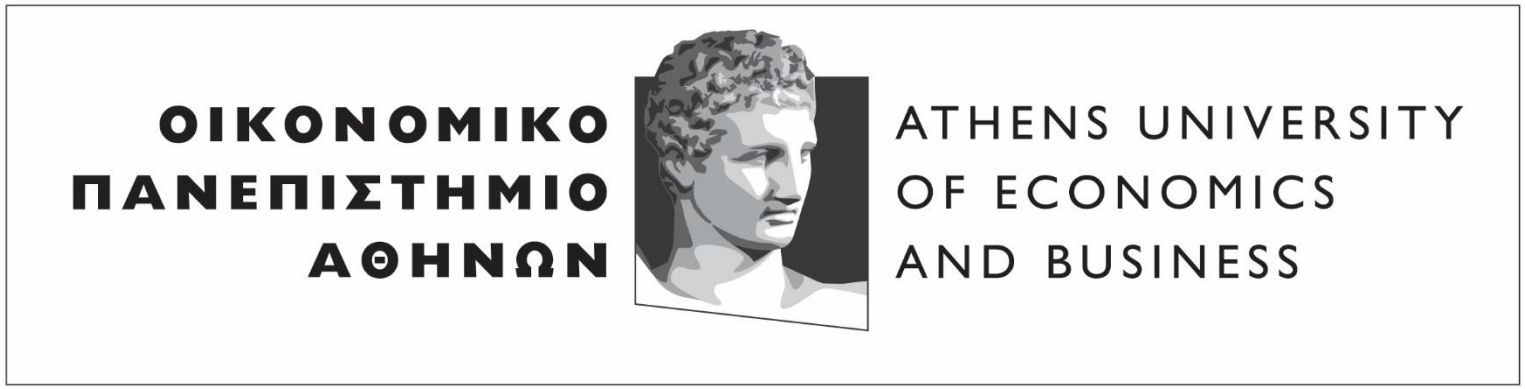
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

πότε μπαίνει στο while()?

Αδιέξοδο (2/2)

```
1.  pthread_mutex_lock(&mutex_1);
2.  while ( pthread_mutex_trylock(&mutex_2) )
    /* Test if already locked   */
3.  {
4.  pthread_mutex_unlock(&mutex_1);  /* Free
    resource to avoid deadlock */
5.  ...
6.  /* stall here   */
7.  ...
8.  pthread_mutex_lock(&mutex_1);
9.  }
10. count++;
11. pthread_mutex_unlock(&mutex_1);
12. pthread_mutex_unlock(&mutex_2);
13. ...
```

http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html



Παράδειγμα: Παράλληλες παραγγελίες

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Παράλληλες παραγγελίες (1 από 3)

- Η παραγγελία περνάει από διάφορα στάδια
  - Πληρωμή, προετοιμασία, ψήσιμο, πακετάρισμα, διανομή
  - Σε ορισμένα βήματα, ένα η περισσότερα νήματα περιμένουν να προσπελάσουν έναν περιορισμένο πόρο
- Παραλλαγή: κρατήσεις εισιτηρίων
  - Τηλεφωνητής, πλάνο, ταμίας

Παράλληλες παραγγελίες (2 από 3)

- Μπορούν να εξυπηρετούνται ταυτόχρονα τόσες παραγγελίες όσοι είναι οι πόροι
  - Απαγορεύεται η ταυτόχρονη τροποποίηση του μετρητή πόρων
  - Δηλαδή ένα νήμα μόνο προσθαφαιρεί τον μετρητή, αν και περισσότερα του ενός μπορούν να εκτελούνται παράλληλα.
  - Μπορεί να πρέπει να δεσμεύσουμε ν πόρους
    - Π.χ. 3 φούρνους για ψήσιμο

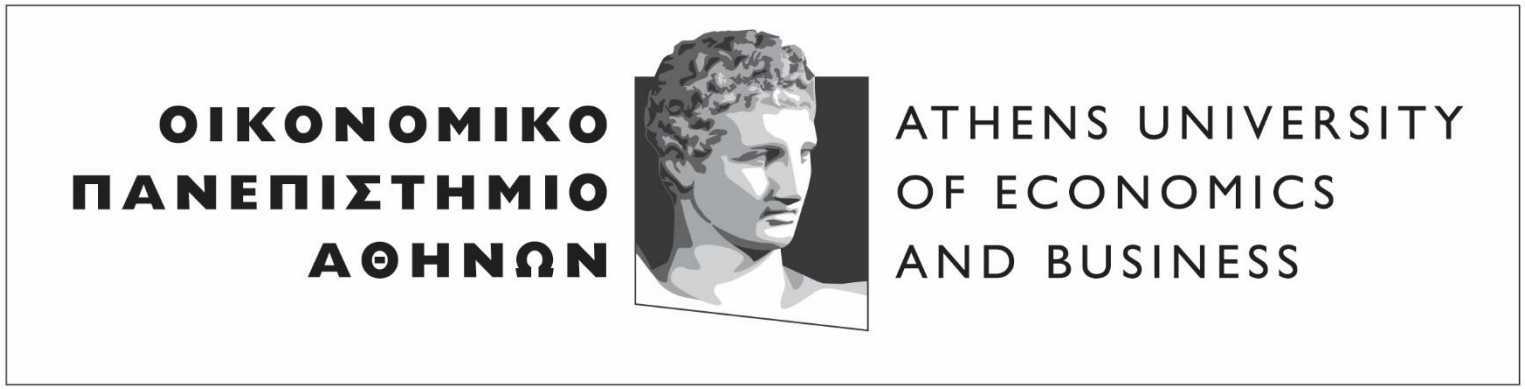
Παράλληλες παραγγελίες (3 από 3)

- Το ζητούμενο είναι ο προσδιορισμός της κρίσιμης περιοχής και ο τρόπος συγχρονισμού των νημάτων χωρίς να μπλοκάρει το ένα το άλλο. Ιδανικά θέλουμε όλες οι παραγγελίες να ανατεθούν (αν υπάρχουν) σε πόρους
- 1 mutex για τον μετρητή διαθέσιμων πόρων
- 1 condition variable για τις παραγγελίες
- Αν θέλουμε να εκτυπώνουν και logs στην κονσόλα;
- Αν δεν περιμένουν όλοι για το ίδιο πλήθος πόρων;

Παράδειγμα με παράλληλες παραγγελίες

```
1.  #include <stdio.h>
2.  #include <pthread.h>
3.  #include <unistd.h>
4.  #define N 10
5.  pthread_mutex_t lock;
6.  pthread_cond_t  cond;
7.  int resources = 2; //diathesimoi poroi
8.  int id[N];
9.  int main()  {
10.    int rc;
11.    pthread_t threads[N];
12.    pthread_mutex_init(&lock, NULL);
13.    pthread_cond_init(&cond, NULL);
14.    for (int i = 0; i < N; i++) {
15.       id[i] = i+1;
16.       printf("Main: dhmioyrgia nhmatos %d\n", i+1);
17.       rc = pthread_create(&threads[i], NULL, order, &id[i]);
18.    }
19.    for (int i = 0; i < N; i++) {
20.       pthread_join(threads[i], NULL);
21.    }
22.    pthread_mutex_destroy(&lock);
23.    pthread_cond_destroy(&cond);
24.    return 0;
25. }
```

```
1.  void *order(void *x){
2.     int id = *(int *)x;
3.     int rc;
4.     printf("H paraggelia %d xekinhse\n",id);
5.     rc = pthread_mutex_lock(&lock);
6.     while (resources == 0) {
7.        printf("H paraggelia %d den brike diathesimo poro.
       Blocked...\n", id);
8.        rc = pthread_cond_wait(&cond, &lock);
9.     }
10.       printf("H paraggelia %d eksipiretitai.\n", id);
11.    resources--;
12.    rc = pthread_mutex_unlock(&lock);
13.    sleep(5); //kane kapoia douleia me ton poro
14.    rc = pthread_mutex_lock(&lock);
15.    printf("H paraggelia %d eksipiretithike epitixos! \n", id);
16.    resources++;
17.    rc = pthread_cond_signal(&cond);
18.    rc = pthread_mutex_unlock(&lock);
19.    pthread_exit(NULL);
20. }
```



Κοινά σφάλματα

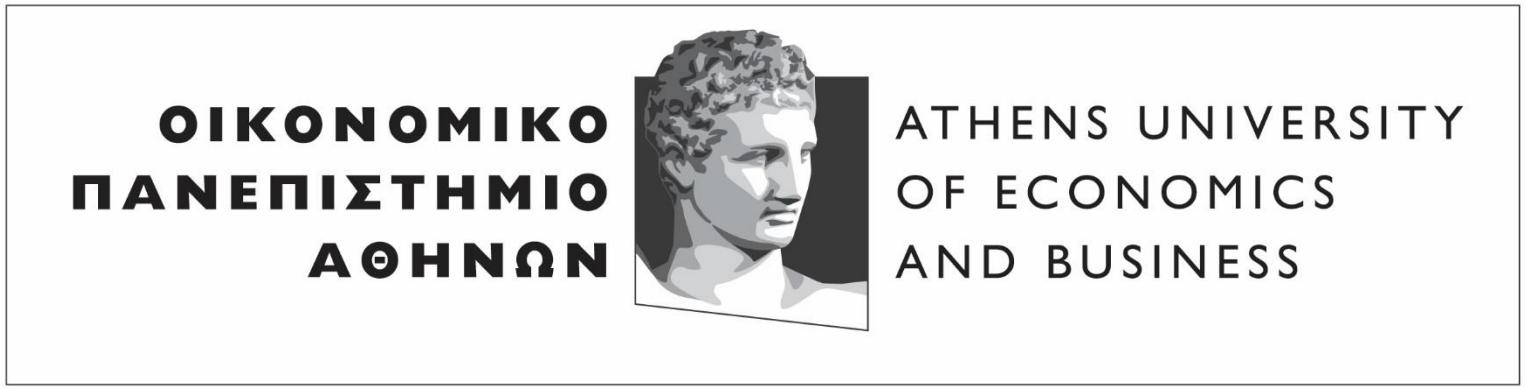
Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Κοινά σφάλματα με νήματα

- Ακρίβεια κλειδώματος – Όσο πιο λεπτομερή είναι τα κλειδώματα, τόσο περισσότερο παραλληλισμό έχουμε, αλλά με μεγαλύτερη επιβάρυνση.
- Κρίσιμες περιοχές– Ελαχιστοποίηση κρίσιμων περιοχών, μπορεί να είναι σημεία συμφόρησης.
- Συχνότητα κλειδωμάτων – Μήπως κλειδώνουμε πολύ συχνά; Μήπως κλειδώνουμε χωρίς να πρέπει; Μείωση κλειδωμάτων για πλήρη αξιοποίηση παραλληλισμού.
- Πολλά νήματα; - Πόσα νήματα είναι πάρα πολλά; Μπορεί να μειώσουν την απόδοση;
- Ακατάστατη έξοδος – Κλειδώσαμε την οθόνη;



Ψευδοτυχαίοι αριθμοί

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Παραγωγή ψευδοτυχαίων αριθμών

- Ψευδοτυχαίοι αριθμοί
  - Παράγονται από μαθηματική συνάρτηση
  - Εξαρτώνται από τον αρχικό αριθμό (σπόρο)
  - Έχουν στατιστικά χαρακτηριστικά τυχαίων
  - Αλλά ο ίδιος σπόρος δίνει την ίδια ακολουθία
- Παραγωγή χωρίς νήματα
  - srand(seed): ορίζει το σπόρο (default: 1)
  - rand(): επιστρέφει επόμενο ψευδοτυχαίο αριθμό

Ψευδοτυχαίοι με νήματα (1/3)

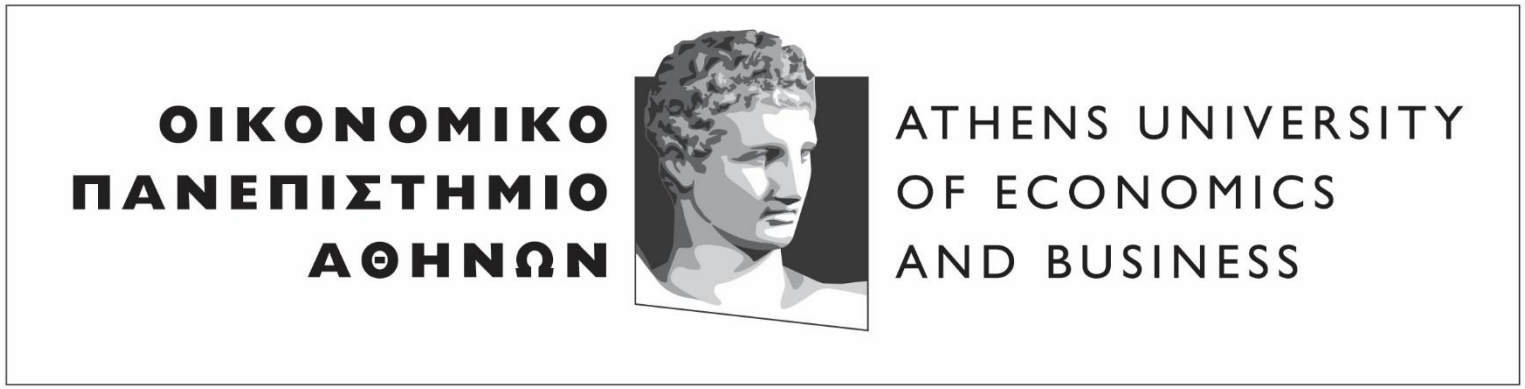
- Η rand() έχει κρυφή κατάσταση
  - Αρχικά έχει 1 ή ότι δώσαμε στη seed()
  - Κάθε φορά που καλείται αλλάζει τον αριθμό
  - Από εκεί προκύπτει ο επόμενος ψευδοτυχαίος
  - Αν έχουμε πολλά νήματα έχουμε πρόβλημα
    - Μοιράζονται όλα την ίδια κρυφή κατάσταση
    - Χωρίς mutex ή κάτι άλλο
    - Ουσιαστικά γράφουν το ένα πάνω στο άλλο

Ψευδοτυχαίοι με νήματα (2/3)

- Λύση: rand_r(&seedp)
  - Η rand_r έχει ως κατάσταση τον seedp
    - Δεν έχει κρυφή κατάσταση
  - Αρχικά seedp = επιθυμητός σπόρος
  - Σε κάθε κλήση τροποποιείται το seedp
  - Κάθε νήμα έχει τη δική του κατάσταση
  - Φτάνει να έχει τη δική του μεταβλητή seedp!

Ψευδοτυχαίοι με νήματα (3/3)

- Πώς να βγάλουμε σωστά ψευδοτυχαίους;
  - Ξεκινάμε με έναν κεντρικό σπόρο
  - Αν τον κάνουμε καθολική μεταβλητή;
    - Όλα τα νήματα θα έχουν την ίδια
    - Ουσιαστικά πάλι γράφουν το ένα πάνω στο άλλο
      - Υπόδειξη: πώς προστατεύουμε καθολικές μεταβλητές;
  - Αν τον αντιγράψουμε σε τοπική μεταβλητή;
    - Τότε κάθε νήμα θα έχει ακριβώς την ίδια ακολουθία
      - Υπόδειξη: αν συνδυάσουμε σπόρο με κάτι από το νήμα;



Βιβλιογραφία

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Βιβλιογραφία και αναφορές

- http://randu.org/tutorials/threads/
- http://www.cs.cf.ac.uk/Dave/C/node30.html
- https://cis.temple.edu/~giorgio/old/cis307f97/readings/pthreads.html
- http://linux.die.net/man/3/

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Τέλος Φροντιστηριακής Ενότητας # 4

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 4 :** Εισαγωγή στα νήματα με POSIX Threads (Μέρος 2ο)

Τμήμα: Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης