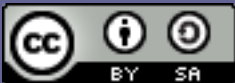


# Παραλληλισμός σε επίπεδο threads

(και το βασικό API των POSIX threads)

<https://mixstef.github.io/courses/parprog/>

Μ.Στεφανιδάκης



# Thread – μια λέξη με πολλές χρήσεις

εφαρμογές και  
threading  
frameworks

work,  
task,  
...

---

user-space  
software

user-level threads,  
fibers,  
...

---

kernel-space  
(OS)

kernel threads,  
threads,  
pthreads, ...

---

hardware

h/w thread,  
execution thread,  
cpu, ...

---

# Τα επίπεδα υλοποίησης των threads

- **Hardware threads**

- Μονάδες εκτέλεσης «ροών» εντολών, με ξεχωριστό **program counter** και **καταχωρητές**
- Ένας πυρήνας (core) ή μέρος του πυρήνα σε περιπτώσεις “hyperthreading” (ή πιο επιστημονικά, “simultaneous multithreading”)

- **OS threads**

- Ελαφριές διεργασίες, μοιράζονται τη μνήμη και τα αρχεία μιας κανονικής διεργασίας
- Το λειτουργικό σύστημα εναλλάσσει την εκτέλεσή τους στα διαθέσιμα hardware threads
- Κάθε thread έχει **δικό του stack**, άρα **ξεχωριστές τοπικές μεταβλητές**

# Τα επίπεδα υλοποίησης των threads

- **User level threads**
  - Υλοποιούνται σε μια διεργασία χωρίς το λειτουργικό σύστημα να έχει γνώση της ύπαρξής τους
  - Κοινό χαρακτηριστικό είναι ο μη προεκτοπισμός: ένα ULT θα δώσει τον έλεγχο σε άλλο μόνο όταν θέλει
  - Εκτελούνται σε ένα η περισσότερα OS threads
- **“Work”, “task”**
  - Είναι κομμάτια εργασίας για την υλοποίηση ενός αλγορίθμου, σε παράλληλη ή εναλασσόμενη εκτέλεση
    - Ανάλογα με τα διαθέσιμα hardware threads
  - Η εφαρμογή ή ένα threading framework θα τα αναθέσει σε user-level και OS threads

# POSIX threads (pthreads)

- **Είναι OS threads**
  - Παρέχονται από το λειτουργικό σύστημα, το οποίο και τα διαχειρίζεται
  - POSIX είναι ένα πρότυπο για την υλοποίηση διάφορων APIs υπηρεσιών των λειτουργικών συστημάτων
  - Κάθε λειτουργικό σύστημα διαθέτει τη δική του υλοποίηση των POSIX threads με τη μορφή κοινής βιβλιοθήκης (shared library)
- **Pthreads API**
  - Παρέχει μεθόδους για τη δημιουργία, τον τερματισμό, τον συγχρονισμό και γενικά τη **διαχείριση των threads** ενός λειτουργικού συστήματος

# Προγραμματισμός POSIX threads

- **Include headers**

- `#include <pthread.h>`
- Ορισμοί του API των POSIX threads (συναρτήσεις, σταθερές και δομές δεδομένων)

- **Compiler flags**

- Π.χ. `gcc -O2 -Wall -pthread one-thread.c -o one-thread`
- Ειδοποιεί τον μεταγλωττιστή (στο στάδιο του linker) να συνδέσει το εκτελέσιμο πρόγραμμά μας με αναφορές στην κοινή βιβλιοθήκη του συστήματος που υλοποιεί τα POSIX threads

- Αν δεν προστεθεί θα δημιουργηθεί σφάλμα παρόμοιο με

```
/usr/bin/ld: /tmp/ccUR1fAh.o: in function `main':  
one-thread.c:(.text.startup+0x27): undefined reference to `pthread_create'  
/usr/bin/ld: one-thread.c:(.text.startup+0x42): undefined reference to `pthread_join'  
collect2: error: ld returned 1 exit status
```

# Η συνάρτηση ενός thread

- Περιέχει την εργασία (work) ενός thread
  - Η συνάρτηση πρέπει να έχει συγκεκριμένη «υπογραφή»
    - Μην ανησυχείτε που δεν επιστρέφεται void \* :-)
  - Το thread εκτελείται μέχρι την κλήση του `pthread_exit()` ή μέχρι να επιστρέψει η συνάρτηση
  - Διαφορετικά threads μπορούν να χρησιμοποιούν την ίδια ή διαφορετικές συναρτήσεις

```
void *thread_func(void *args) {  
  
    // useful work here  
    printf("Child thread working..\n");  
  
    // terminate and let be joined  
    pthread_exit(NULL);  
}
```

# Δημιουργία ενός thread: pthread\_create()

- **Ορίσματα**

- δείκτης σε pthread\_t: επιστρέφεται μια τιμή «χειρισμού» του thread – θα χρησιμοποιηθεί σε επόμενες συναρτήσεις για το thread αυτό
- δείκτης σε δομή με ειδικά χαρακτηριστικά (προτεραιότητα, μέγεθος stack κλπ) για το νέο thread, NULL = defaults
- δείκτης στη συνάρτηση του thread
- δείκτης σε παραμέτρους που θα περάσουν στη συνάρτηση του thread

```
pthread_t pid;    // the thread's "handle"

// create a thread - non-blocking call
if (pthread_create(&pid, NULL, thread_func, NULL) != 0) {
    printf("Error in thread creation!\n");
    exit(1);
}
```



# Αναμονή τερματισμού ενός thread: pthread\_join()

- **Ορίσματα**

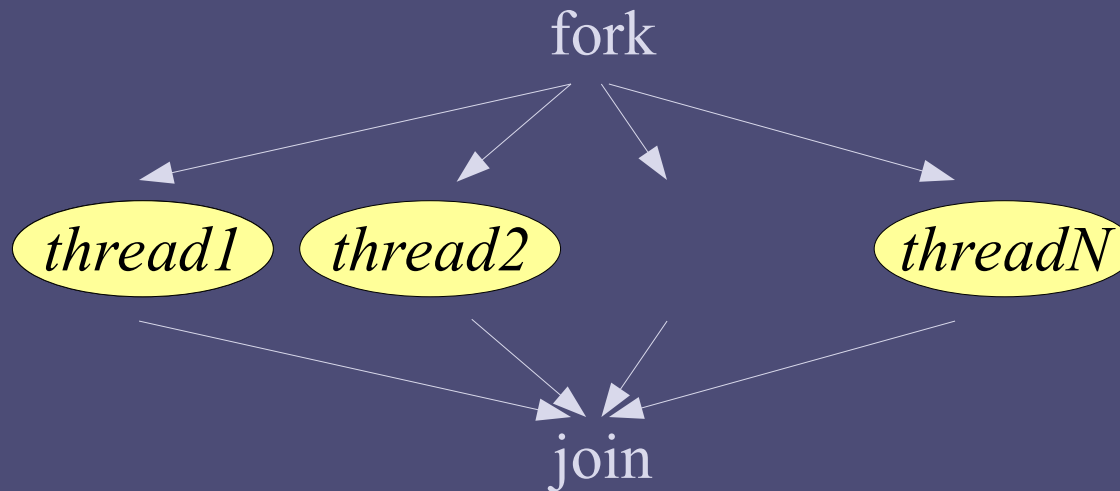
- η τιμή «χειρισμού» του thread που περιμένουμε να τελειώσει – αν αυτό έχει ήδη γίνει, η pthread\_join() επιστρέφει αμέσως
- δείκτης σε θέση υποδοχής της τιμής που επέστρεψε το thread στον τερματισμό (μέσω της pthread\_exit() ή με return) – NULL = δεν χρησιμοποιείται

```
// wait until thread terminates - blocking call
if (pthread_join(pid, NULL) != 0) {
    printf("Error in thread join!\n");
    exit(1);
}
```

# Τρόπος χρήσης των `pthread_create()` και `pthread_join()`

- Το σχήμα “fork – join”

- Ένα “master thread” (π.χ. το «κυρίως πρόγραμμα») δημιουργεί (“fork”) με την `pthread_create()` μια σειρά από threads για να εργαστούν παράλληλα και περιμένει να ολοκληρώσουν την εργασία τους (“join”) με την `pthread_join()`



# Παράδειγμα “fork – join”

πίνακας με handles

```
pthread_t pid[THREADS];    // the threads' "handles"

for (int i=0;i<THREADS;i++) {
    // create thread i - non-blocking call
    if (pthread_create(&pid[i],NULL,thread_func,NULL)!=0) {
        printf("Error in thread creation!\n");
        exit(1);
    }
}

// useful work here
printf("Main thread working..\n");

for (int i=0;i<THREADS;i++) {
    // wait until thread i terminates - blocking call
    if (pthread_join(pid[i],NULL)!=0) {
        printf("Error in thread join!\n");
        exit(1);
    }
}
```

# Παραμετροποίηση threads

- Πώς κάθε thread θα εκτελέσει μέρος της συνολικής εργασίας
  - με την ίδια συνάρτηση thread!
  - Παραμετροποίηση με τη βοήθεια ορίσματος στη συνάρτηση του thread

```
// struct of info passed to each thread
struct thread_params {
    int id; // thread's id (for demo purposes)
};

void *thread_func(void *args) {
    // get arguments
    struct thread_params *tp = (struct thread_params *)args;
    int id = tp->id;

    // useful work here
    printf("Child thread %d working..\n", id);
}
```

*δομή ως «πακέτο» παραμέτρων*

*μετατροπή (cast) από void ptr → struct thread\_params ptr*

*χρήση παραμέτρων*

# Παραμετροποίηση threads

- Σε κάθε thread περνάει ένα διαφορετικό «πακέτο» κατά την `pthread_create()`
  - Οι δομές «πακέτα» θα πρέπει να είναι διαφορετικές για κάθε thread
  - Και να διατηρούνται σε όλη τη διάρκεια ζωής του thread

```
struct thread_params tparam[THREADS]; // array of info structs
// (one per thread)
// for all threads
for (int i=0; i<THREADS; i++) {
    // fill info of tparam[i]
    tparam[i].id = i;
    // create i-th thread, pass ptr to tparam[i]
    if (pthread_create(&pid[i], NULL, thread_func, &tparam[i]) != 0) {
        printf("Error in thread creation!\n");
        exit(1);
    }
}
```

*πίνακας με «πακέτα» παραμέτρων*

*αρχικοποίηση «πακέτου» για thread i*

*πέραςμα στην pthread\_create()*