ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Λειτουργικά Συστήματα

Φροντιστηριακή ενότητα #2 : Διαχείριση μνήμης και διεργασίες

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

- Διαχείριση της μνήμης με έμφαση στα πιθανά λογικά λάθη (Memory leaks, κ.ο.κ.)
- Χρήση δεικτών σε πίνακες, πέρασμα παραμέτρων, κ.ο.κ.
- Διεργασίες και συναφείς χρήσιμες κλήσεις συστήματος

Περιεχόμενα ενότητας

- Δείκτες και θέματα διαχείρισης μνήμης
- Διεργασίες

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Δείκτες και θέματα διαχείρισης μνήμης

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 2:** Προγραμματισμός με τη γλώσσα C για γνώστες της γλώσσας Java

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

Μεταβλητές και δείκτες

- https://www.thegeekstuff.com/2012/01/advanced-c-pointers/
- Κάθε μεταβλητή έχει
  - Τη τιμή της και τη διευθυνση της μνήμης που της αντιστοιχεί
- Δείκτης (pointer): μεταβλητή που έχει ως τιμή τη διευθυνση μιας άλλης μεταβλητής
  - Ο τύπος του δείκτη ταιριάζει με εκείνον της μεταβλητής στην οποία δείχνει.
  - int a = 10; int * a_ptr = &a;

Τελεστές δεικτών

& τελεστής που επιστρέφει θέση μνήμης

```
int i = 10;

int* ptr = &i; // περιεχόμενο της ptr είναι η θέση μνήμης του i
```

***** τελεστής που δείχνει περιεχόμενο στη θέση μνήμης

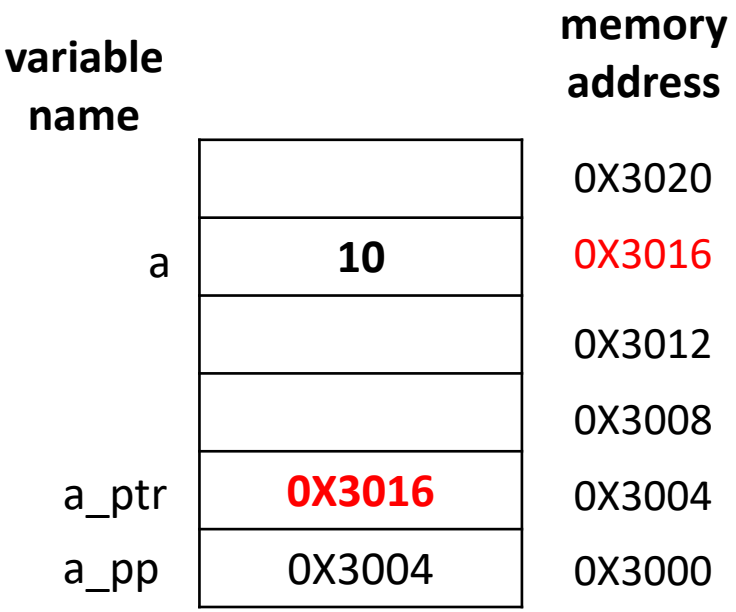
```
int j = *ptr; // το j παίρνει τιμή από τη θέση μνήμης που δείχνει ο ptr
```

Παραδείγματα δεικτών

- unsigned int *ui_ptr;
- char* char_ptr;
- int **ptr2ptr; // δείκτης σε δείκτη
- struct *strct_ptr;
- int i = 1;
  - int *ptr = &i;
  - int j = *ptr; // ανάκτηση και ανάθεση τιμής του “i”
  - *ptr = 43; // το “i” ισούται τώρα με 43

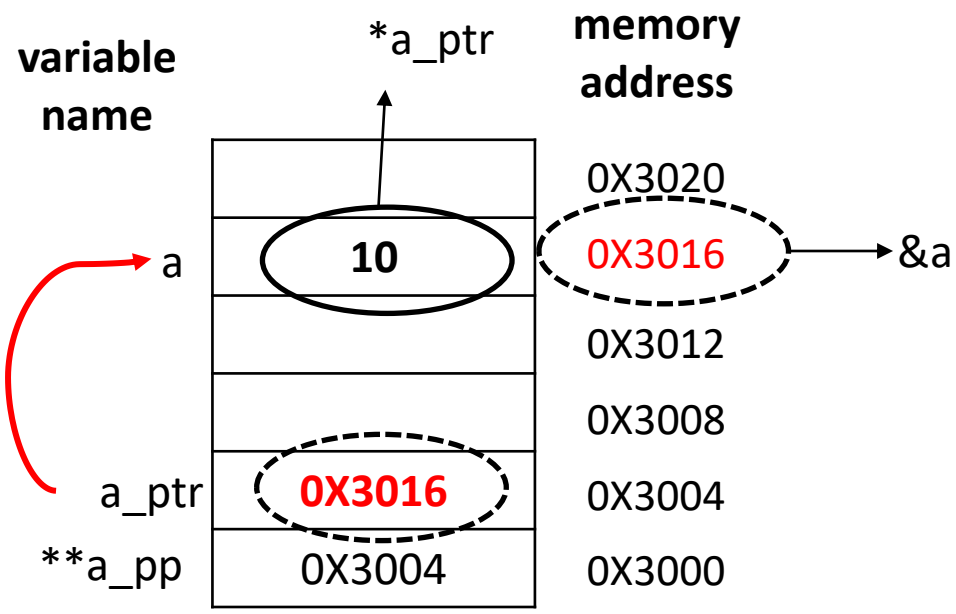
Μεταβλητές και δείκτες

a =	
&a =	
*a_ptr=	
&a_ptr =	
a_ptr =	
a_pp =	
**a_pp =	
a_ptr++;	
a_ptr =	



Μεταβλητές και δείκτες

a =	10
&a =	0X3016
*a_ptr=	10
&a_ptr =	0X3004
a_ptr =	0X3016
a_pp =	0X3004
**a_pp =	10
a_ptr++;	
a_ptr =	0X3020



Πέρασμα παραμέτρων κατά τιμή

```
#include <stdio.h>

void swap(int x,int y){
  int temp = x;
  x = y;
  y = temp;
}
```

```
int main(void) {
  int x = 9;
  int y = 5;
  swap(x, y);
  printf("x = %d,y = %d\n", x,y);
  return 0;
}
```

Πέρασμα παραμέτρων κατά αναφορά

```
#include <stdio.h>

Void swap(int*x,int*y){
  int temp = *x;
  *x = *y;
  *y = temp;
}
```

```
int main(void) {
  int x = 9;
  int y = 5;
  swap(&x, &y);
  printf("x = %d,y = %d\n", x,y);
  return 0;
}
```

Δείκτες και structs

```
#include <stdio.h>
struct person{
   int age;
   float weight;
};
int main(){
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter age: ");
    scanf("%d", &(personPtr->age));
    printf("Enter weight: ");
    scanf("%f", &(personPtr->weight));
    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f\n", personPtr->weight);
    return 0;
}
```

Πίνακες και δείκτες

```
// C program to illustrate pointers in arrays
#include<stdio.h>
int main() {
  int x[5] = {1, 2, 3, 4, 5};
  int* ptr;

  // ptr is assigned the address of the third element
  ptr = &x[2];

  printf("*ptr = %d \n", *ptr);   // 3
  printf("*(ptr+1) = %d \n", *(ptr+1)); // 4
  printf("*(ptr-1) = %d\n", *(ptr-1));  // 2

  return 0;
}
```

*ptr = 3
*(ptr+1) = 4
*(ptr-1) = 2

Πίνακας ⇔ σταθερός δείκτης

- &arr[0] ⇔ arr
- int * p;
- arr = p;
  - Λάθος! Ο arr είναι const δείκτης
  - Δεν μπορεί να δείξει σε άλλη διεύθυνση μνήμης
- void foo(int arr[], int len) ⇔ void foo(const int * arr, int len)

C Function Pointers (1/2)

<return type> (*<pointer>)
(arguments)

Πχ: int (*fptr)(int, int);

```
void f(void (*a)() ) {
    a(); // καλεί ό,τι περαστεί ως a
}
void test() {
    printf("hello world\n");
}
int main() {
   f(&test);
   return 0;
}
```

C Function Pointers (2/2)

```
int max(int a, int b) {
  return (a > b ? a : b);
}
int foo(int a, int b, int(*func)(int p1, int p2)){
  return func(a, b);
}
int main(void) {
  int result = foo(11, 22, &max);
  printf("%d\n", result);
  return 0;
}
```

Memory Leaks (1/2)

```
1.  int main() {
2.    int *i = (int*)malloc(3*sizeof(int));
3.    i = 0; /*Wxxxxx, EXASA TON pointer STH
                DYNAMIKA DESMEYMENH MNHMH*/
4.    free(???); /* DE KSERW PLEON PWS..*/
5.  }
```

Memory Leaks (2/2)

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  int main() {
4.      int *array = (int *) malloc(10*sizeof(int));
5.      if (array == NULL) {
6.          printf("Out of memory!\n");
7.          return -1;
8.      }
9.      free(array);
10.     return 0;
11. }
```

Διαχείριση δυναμικής μνήμης

Function	Description
malloc	**allocates the specified number of bytes**
realloc	increases or decreases the size of the specified block of memory. Reallocates it if needed
calloc	allocates the specified number of bytes and initializes them to zero
free	**releases the specified block of memory back to the system**

- https://en.wikipedia.org/wiki/C_dynamic_memory_allocation

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Διεργασίες

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 2:** Προγραμματισμός με τη γλώσσα C για γνώστες της γλώσσας Java

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

Εισαγωγικές έννοιες

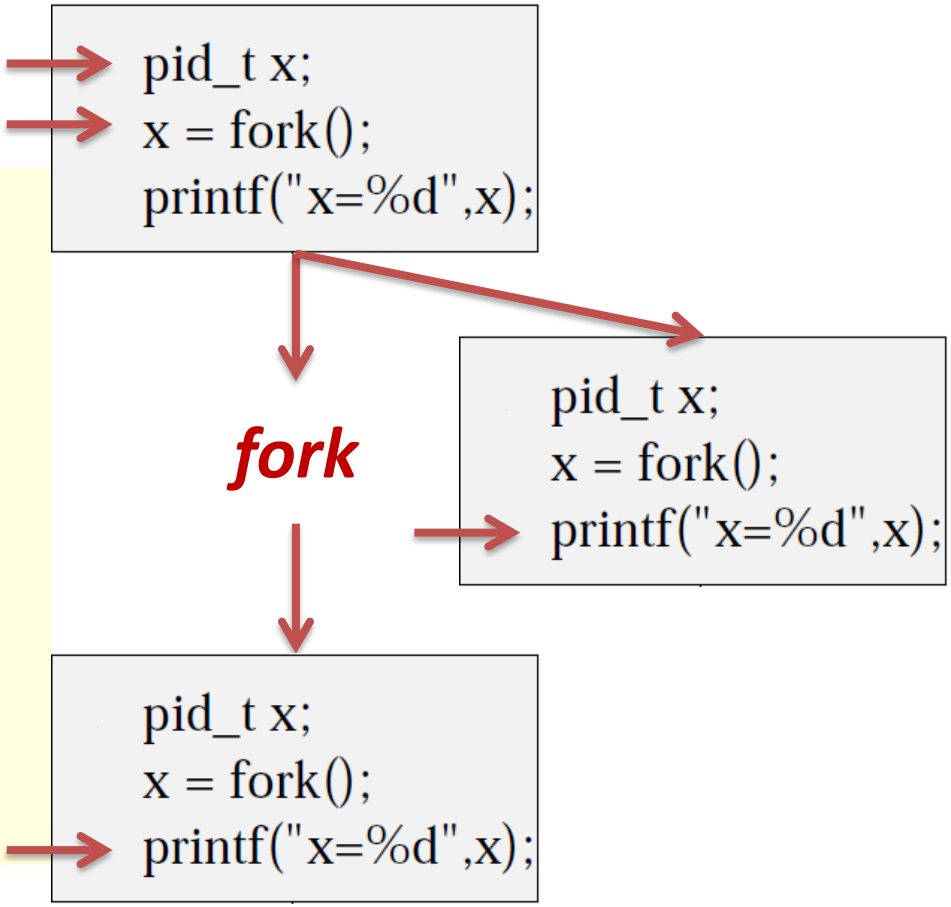
- Διεργασία: πρόγραμμα σε εκτέλεση
- Η μία διεργασία δεν επηρεάζει άμεσα την άλλη (isolation)
- Shared memory?
  - Καλύτερα Νήματα αν πρέπει να μοιραστούν μνήμη
  - Δεν χρειάζεται να παρεμβάλλεται ο πυρήνας

Δημιουργία Διεργασίας (1/4)

- Μια διεργασία δημιουργείται από μια άλλη
  - Π.χ. από το shell ανοίγω μια διεργασία gedit
  - «γενικό πλαίσιο» (context):
    - PID – process ID
    - Μνήμη: Πρόγραμμα εντολών + δεδομένα
    - Μετρητής προγράμματος – program counter
    - Χειριστές σημάτων – signal handlers
      - Signals: https://en.wikipedia.org/wiki/Signal_(IPC)
    - …

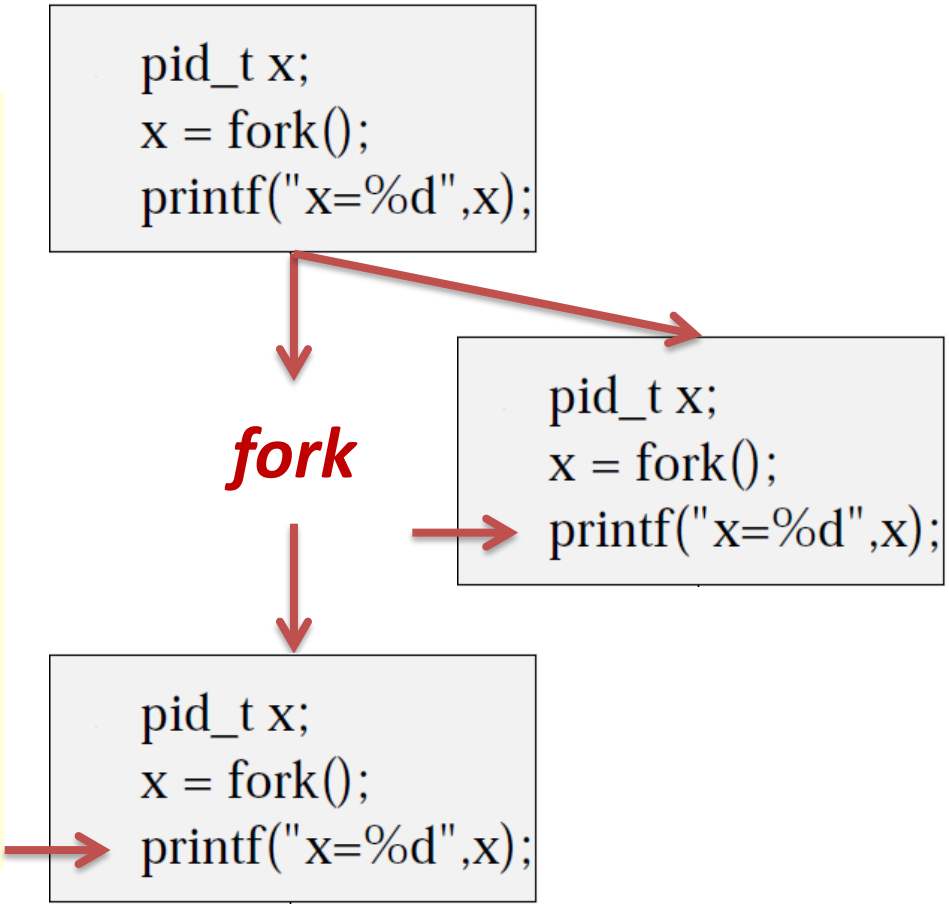
Δημιουργία Διεργασίας (2/4)

```
1.  #include <sys/types.h>
2.  #include <unistd.h>
3.  pid_t fork(void);
       /* Δημιουργεί διεργασία
    παιδί ⇔ ακριβές αντίγραφο
    της διεργασίας από το σημείο
    εκτέλεσης και μετά */
```

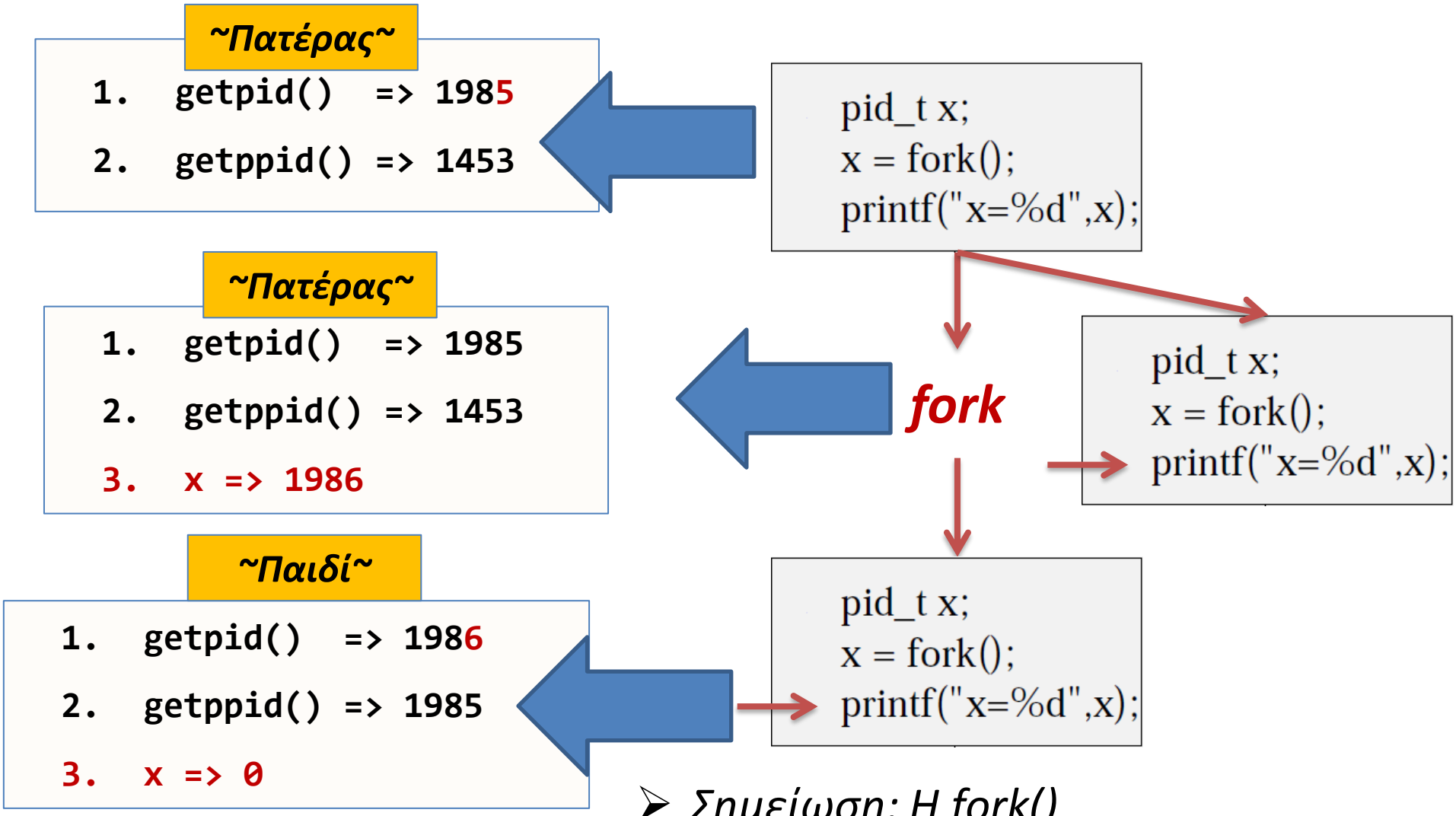


Δημιουργία Διεργασίας (3/4)

```
1.  #include <sys/types.h>
2.  #include <unistd.h>
3.  pid_t getpid(void);
       // μαθαίνεις την PID σου
4.  pid_t getppid(void);
       // PID του πατέρα σου
```



Δημιουργία Διεργασίας (4/4)



- *Σημείωση: Η fork() επιστρέφει -1 αν αποτύχει*

Τι κάνει το ακόλουθο;

```
1.  pid_t pid;
2.  pid = fork();
3.  if (pid<0) {
4.       perror(“LA8OS STHN fork()");
5.       exit(1);
6.  }
7.  if (!pid) // ισοδύνο του (pid==0)
8.     printf("I am the child process of my par
    %d\n“,getppid());
9.  else if(pid) // if (pid!=0)
10.    printf("I am the parent process of %d\n“, pid);
```

Zombies!

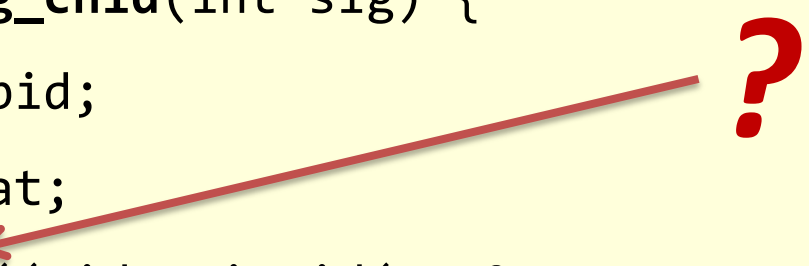
- Zombie status: μια διεργασία που τελείωσε
  - Δεν διαγράφεται από το σύστημα
  - Παραμένει έως ο πατέρας να λάβει το termination status
- Αν ο πατέρας πεθάνει νωρίτερα;
  - Υιοθετούνται από το process με PID 1 (init)
  - Γίνονται ζόμπι και τρώνε τους πόρους του συστήματος
    - Καταχωρήσεις στον πίνακα διεργασιών!
- Αποφυγή ζόμπι: ο πατέρας περιμένει ρητά με τη wait()
  - Εναλλακτικά φτιάχνει συνάρτηση signal handler για το SIGCHLD

Χρήση της wait

```
1. #include <sys/types.h>
2. #include <sys/wait.h>
3. pid_t wait(int *status);/* PERIMENEI
     * OPOIODHPOTE PAIDI. PAIRNEI POINTER
     * GIA NA EPISTREPSEI STATUS */
4. // pid_t waitpid(pid_t pid /*-1 opoios-
   //    dhpote*/, int *status, int options);
   // option=WNOHANG makes the call return
   // immediately if no children completed;
   // otherwise, use option=0
```

Εναλλακτικά: **SIGCHLD** signal handler

```
1.  void sig_chld(int sig) {
2.   pid_t pid;
3.   int stat;
4.   while ((pid=waitpid(-1,&stat,WNOHANG))>0){
5.     printf("Child %d terminated with status
                                  %d\n",pid,stat);
6.   }
7.   signal(SIGCHLD,sig_chld);/* DHLWNOUME POIA signal
    handler SYNARTHSH 8A KLH8EI APO TO PROGRAMMA MAS..*/
8.  }
```



https://www.thegeekstuff.com/2012/03/catch-signals-sample-c-code/

fork() + exec()

```
1. pid_t pid=fork();
2. if (pid==0) // AN EISAI TO PAIDI
3.   exec(“./a.out"); // PATH PROS hello?
4.   perror("Error in exec!\n");
5.   exit(1);
6. }
```

execl ή execv | execlp ή execvp

```
// execl()
// execlp() @ $PATH
1. execl("/bin/ls",
         "ls", "-l",
        "/home/csuser",
               NULL);
```

```
// execv()
// execvp() @ $PATH
1. char *params[4]=
       {"ls", "-l",
      "/home/csuser",
             NULL};
2. execv("/bin/ls",
           params);
```

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Τέλος Φροντιστηριακής Ενότητας # 2

Μάθημα: Λειτουργικά Συστήματα, **Φροντιστηριακή Ενότητα # 2:** Προγραμματισμός με τη γλώσσα C για γνώστες της γλώσσας Java,
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