HY255 Εργαστήριο Λογισμικού – L01

Εισαγωγή στο HY255

Επανάληψη στη C

HY255 Εργαστήριο Λογισμικού
Άνοιξη 2024
Άγγελος Μπίλας

HY255 Εργαστήριο Λογισμικού

- Συστήματα λογισμικού
  - Πολλά και σημαντικά
  - Και μάλιστα όλο και περισσότερα
  - Και δυστυχώς όλο και πολυπλοκότερα
- “These systems make the world run”
  - Εφαρμογές στο cloud
  - Εφαρμογές σε κινητά συστήματα
  - Λειτουργικά συστήματα
  - Κατανεμημένα και παράλληλα συστήματα
  - Ενσωματωμένα συστήματα

Τι είναι δύσκολο στα συστήματα λογισμικού;

- Για κάποιο λόγο η πολυπλοκότητά τους αυξάνεται διαρκώς
- Παρατηρείστε ιστορικά
  - Μικρές συσκευές κάνουν όλο και περισσότερες δουλειές (αντί να απλοποιούνται)
  - Τα αυτοκίνητα μετατρέπονται σε μεγάλο βαθμό σε συστήματα λογισμικού (και υπηρεσιών)
  - Θα δυσκολευτεί κανείς να βρει κάτι που έγινε «απλούστερο» με τον χρόνο
- Από την άλλη πλευρά δεν υπάρχουν αυτόματοι τρόποι να σχεδιάζουμε τέτοια συστήματα
  - Πολύ θα θέλαμε να παράγουμε τα προγράμματα στα οποία βασιζόμαστε με αυτόματο τρόπο, π.χ. με κάποιο «υπερ-πρόγραμμα»
  - Αδύνατο να γίνει αυτό π.χ. για ένα λειτουργικό σύστημα και εξαιρετικά δύσκολο ακόμη και για απλά συστήματα λογισμικού που παρέχουν μια “νέα” λειτουργία
- Πρέπει εμείς να εκπαιδευτούμε να σχεδιάζουμε σωστά τέτοια συστήματα
  - Να κατανοούμε πως δουλεύουν
  - Να μπορούμε να τα υλοποιήσουμε με τρόπο που να δουλεύουν
  - Να βελτιώνουμε τα εργαλεία που έχουμε στη διάθεσή μας

Ο Κόσμος μας στο ΗΥ255

- Μπορούμε να χωρίσουμε τον κόσμο σε περίπου τρία επίπεδα
- 1. Λογισμικό επεξεργασίας δεδομένων
  - πχ scripting languages, ίσως python, etc.
- 2. Λογισμικό εφαρμογών
  - Πχ web servers, σε μεγάλο βαθμό Java και παρεμφερείς γλώσσες (με managed runtime systems)
- **3. Λογισμικό συστημάτων → ΗΥ255**
  - **Λειτουργικά συστήματα, compilers, πρωτόκολλα δικτύων, συστήματα δεδομένων, παράλληλα συστήματα, κοκ.**
  - **Σε μεγάλο βαθμό C (αλλά όχι μόνο)**

HY255

- (1) Design/Compile-time: Πως σχεδιάζουμε συστήματα λογισμικού
  - (Ευτυχώς) Τα πολύπλοκα συστήματα αποτελούνται από άλλα, πιο απλά
  - Πως χωρίζουμε ένα μεγάλο σύστημα/πρόγραμμα σε μικρότερα τμήματα;
  - Πως σχεδιάζουμε με ακρίβεια απλά υποσυστήματα αποτυπώνοντας σωστά ότι έχουμε κατά νου;
  - Πως τα συνθέτουμε σε μεγαλύτερα συστήματα;
  - Ποια είναι η αντιστοιχία των μηχανισμών μια γλώσσας με τους μηχανισμούς που παρέχει ένα σύστημα;
- (2) Runtime: Τι συμβαίνει κατά τον χρόνο εκτέλεσης ενός προγράμματος;
  - Που βρίσκονται τα δεδομένα του προγράμματος;
  - Μπορεί να υπάρχει εκτελέσιμος κώδικας στη στοίβα;
  - Πως λειτουργεί το heap;
- (3) Tools: Τι (ουσιαστικά) εργαλεία έχουμε στη διάθεσή μας για να σχεδιάσουμε και να υλοποιούμε συστήματα λογισμικού;
  - Ποιος είναι ο ρόλος των assertions στο debugging;
  - Είναι απαραίτητο το incremental compilation;
- Στόχος: Να μπορείτε να υλοποιήσετε το επόμενο σύστημα λογισμικού χαμηλού επιπέδου, πχ λειτουργικό σύστημα, compiler, πρωτόκολλο δικτύου, παράλληλο σύστημα.

Το Μάθημα

- http://www.csd.uoc.gr/~hy255
  - Διαβάστε τις πληροφορίες του μαθήματος
  - Διαβάστε το section με τα policies
  - Τις περισσότερες φορές θα συμβουλεύεστε το syllabus
- Διαλέξεις
  - Remote για αυτό το εξάμηνο
  - Παρακολουθείτε + συμμετέχετε με ερωτήσεις/απαντήσεις
  - Αντιστοιχία της ύλης σε δύο από τα βιβλία που βρίσκονται στην βιβλιοθήκη
  - Μπορείτε να συμβουλευτείτε και άλλα βιβλία που νομίζετε σας εξυπηρετούν καλύτερα
  - Summaries σε slides, όπως αυτά
- 6 ασκήσεις / εργαστήρια
  - Μια άσκηση περίπου ανά δύο εβδομάδες
  - (εκτός από την εβδομάδα των προόδων)
  - 1st Assignment due in two weeks
  - Ερωτήσεις σε forum e-learn
  - Χρησιμοποιήστε τα office hours των βοηθών
- Πρόοδος – τελικό διαγώνισμα
  - Ανάλογα με το πως εξελίσσονται τα της επιδημίας
- Βοηθοί
  - Office hours
  - Κάντε ερωτήσεις, μιλήστε μαζί τους για τον κώδικά σας, εκμεταλλευτείτε τον χρόνο που έχουν να αφιερώσουν

Επανάληψη στη Γλώσσα C

- Από ποια βασικά τμήματα αποτελείται ένας υπολογιστής;
  - Επεξεργαστής
  - Μνήμη
  - Περιφερειακά
- Οι γλώσσες αποτελούν μια αφαίρεση αυτών των τμημάτων, ώστε να μας διευκολύνουν
  - Εντολές, τελεστές → Επεξεργαστής
  - Μεταβλητές → Μνήμη
  - Βιβλιοθήκες → Περιφερειακά

Statements and Operators

- Μας βοηθούν να περιγράψουμε τις πράξεις που θέλουμε να κάνουμε

Statements

- Assignment: x=1;
- Control flow: if-then-else, etc.
- Loops: for, while, etc.
- Function calls: f(x);
- That's pretty much it. For most languages.

Operators

- Arithmetic: +, *, /, --, ...
- Relational: <, <=, ==, ...
- Logical: &&, ||, !
- Bitwise: &, |, ~, <<, >>, ...
  - Also called boolean
- Array indexing: []
- Structure indexing: .
- Pointers: &, *
- Weird
  - , expression separator
  - ; statement terminator
- **And we have to worry about precedence of operators**

Μεταβλητές στη C

- Global και Local (όπως θα δούμε αργότερα)
- Περιέχουν τα δεδομένα του προγράμματός μας: έναν αριθμό, ένα string, κοκ
- **Κάθε** μεταβλητή έχει **έναν** τύπο
- Ο τύπος μιας μεταβλητής ορίζει δύο πράγματα
  - 1. Το μέγεθος της μεταβλητής σε bytes στη μνήμη
  - 2. Τις πράξεις που επιτρέπεται να κάνουμε με την μεταβλητή

Τύποι δεδομένων στη C

- Βασικοί τύποι (σε συστήματα 32 bit)
  - char → guaranteed to be 1 byte in all language implementations
  - int → συνήθως 4 bytes
  - short → συνήθως 2 bytes
  - long → συνήθως 8 bytes
  - float → συνήθως 4 bytes
  - double → συνήθως 8 bytes
  - (unsigned int, unsigned short int, unsigned long int, unsigned char)
- Structured (δομημένοι) τύποι
  - Arrays
  - Structures
  - Enumerated
  - Pointers: Θα τους δούμε σε λεπτομέρεια σε επόμενο μάθημα
  - Unions: Δεν θα πούμε κάτι, αν θέλετε δείτε σε κάποιο από τα βιβλία την χρήση τους (και τα προβλήματά τους)

Arrays

- Ένα σύνολο από στοιχεία του ίδιου τύπου
  - int A[10]: Το A είναι ένα array 10 θέσεων από integers
  - Κατά σύμβαση οι δείχτες των στοιχείων στο array ξεκινούν από το 0, οπότε είναι 0..9
- **Μέγεθος:** Το άθροισμα των μεγεθών των στοιχείων, πχ το A είναι 10 * 4 = 40 bytes (συνήθως, σε συστήματα με 32 bit πράξεις/διευθύνσεις)
- **Πράξεις:** Στα arrays επιτρέπεται μόνο μία πράξη, το indexing [] για να πάρουμε ένα στοιχεί του
  - Πχ A[3] είναι το 4ο στοιχείο του array
- Initialization (convenience)

```
int A[3] = {0,0,0};
int A[3] = {0}; /* compiler infers 0 for rest of elements */
int A[] = {0,0,0}; /* compiler infers array size */
```

- Πολυδιάστατα arrays: Arrays που κάθε στοιχείο τους είναι ένα άλλο array
  - float B[2][3]: Το B είναι ένα array 2 θέσεων από arrays 3 θέσεων από floats

```
int B[2][3] = { {1,2,3},
                {3,4,5}}
```

- Τα arrays τοποθετούνται στη μνήμη με τις σειρές τους (rows) να είναι σε συνεχόμενες θέσεις μνήμης
- Τα arrays έχουν κάποια στενή σχέση με τους pointers
  - θα το δούμε σε λεπτομέρεια σε επόμενο μάθημα

Structures

- Ένα σύνολο από τιμές διαφορετικών τύπων
  - Τα στοιχεία ενός struct έχουν ονόματα (σε αντίθεση με τα arrays που έχουν index)

```
struct {
    char n[32];
    int age;
} person, you;
```

  - Το person είναι μεταβλητή τύπου struct με δύο πεδία, ένα array 32 χαρακτήρων που ονομάζεται n, και έναν integer που ονομάζεται age
- **Μέγεθος:** Το άθροισμα του μεγέθους των επιμέρους πεδίων (συνήθως)
  - Τα structures μπορεί να έχουν μεγαλύτερο μέγεθος από το απλό άθροισμα για λόγους απόδοσης (πχ alignment με τις caches)
- **Πράξεις:** Επιτρέπονται δύο πράξεις
  - 1. "." που επιστρέφει ένα (named) πεδίο της μεταβλητής τύπου structure

```
person.age = 18;
```

  - 2. "=" που αναθέτει όλα τα περιεχόμενα μιας μεταβλητής τύπου structure σε μια άλλη μεταβλητή του ίδιου τύπου

```
you = person; /* Αντιγράφει τα περιεχόμενα του struct person στο struct you */
```

- Προσοχή: Δεν έχουμε πει ακόμη τίποτε για pointers
  - Τα structures και η χρήση τους (οι πράξεις που επιτρέπονται) δεν συνδέονται με τους pointers

Enumerated Type

- Ένα σύνολο από σταθερές τιμές (που επομένως δεν αλλάζουν)

```
enum {RED, BLACK, WHITE} c;
enum color {RED, BLACK, WHITE} c;

enum color {RED, BLACK, WHITE};
enum color c;
```

- Οι enum values/identifiers (RED, BLACK, …) είναι int constants

```
enum {RED=1, BLACK=10, WHITE=100} c;
enum {RED=1, BLACK, WHITE} c;
```

- Οι μεταβλητές τύπου enum μπορεί να πάρουν τιμές μόνο από αυτές που ορίζει ο τύπος
  - Το c είναι μια μεταβλητή τύπου enum που μπορεί να πάρει μόνο τις τιμές RED, BLACK, WHITE
- **Μέγεθος:** Μια μεταβλητή τύπου enum πιάνει τόσο χώρο στη μνήμη όσο χρειάζεται για να αναπαρασταθούν οι τιμές του τύπου
  - Πχ για τρεις τιμές χρειαζόμαστε 2 bit, άρα το c θα μπορούσε να είναι 2 bit (ανεξάρτητα από τις τιμές των constants που είναι int και επομένως 4 bytes η κάθε μια)
  - Ωστόσο, για λόγους απόδοσης συνήθως αυτό «στρογγυλεύεται» σε bytes, άρα το c θα χρειαστεί 1 byte
  - Ανάλογα με την υλοποίηση της γλώσσας, η στρογγύλευση μπορεί να γίνεται και σε words
- **Πράξεις:** Επιτρέπονται δύο πράξεις σε μεταβλητές τύπου enum
  - 1. Ανάθεση με το "=", π.χ. c = RED;
  - 2. Σύγκριση με σταθερές του τύπου, π.χ. if (c = = BLACK)
  - Άρα αν γράψουμε c = YELLOW; είναι compile-time error, όπως και το if (c == YELLOW)
- Οι τύποι enum αποτελούν μια σημαντική βοήθεια για να εκφράζουμε τα προγράμματά μας με ακρίβεια σε ότι αφορά σύνολα από σταθερές τιμές
  - Γιατί απλά δεν χρησιμοποιούμε τον τύπο integer για τις μεταβλητές τύπου enum;

Constants και Constant Expressions

- Οι σταθερές (constants) έχουν τύπους
  - char
    - 'a' character a
    - '\035' character with octal code 035
    - '\x1a' character with hex code 1a
    - '\t' tab
    - '\n' newline
    - '\0' character with code 0 (null char=NUL, but not NULL)
  - int
    - 128 decimal 128
    - 0156 octal 156
    - 0x1abc hexadecimal 0x1abc
  - long
    - 128l (do not use this)
    - 128L
  - float
    - 15.5f, 3.14e2f
    - 15.5F, 3.14e2f
  - double
    - 15.5 η έλλειψη προσδιοριστικού τύπου σημαίνει double
    - 15.5l, 3.14e2L
    - 15.5L, 3.14e2L
- Constant expressions είναι εκφράσεις που η τιμή τους μπορεί να υπολογιστεί την ώρα της μετάφρασης (at compile time)

```
const int K = 1024;
const int M = 1024*1024;

#define PAGESIZE 4096
char A[PAGESIZE*100];
```

sizeof

- Επιστρέφει το μέγεθος ενός τύπου ή μεταβλητής στην συγκεκριμένη υλοποίηση της γλώσσας
  - Πχ intsize = sizeof(int);
- Είναι ένα δεσμευμένο keyword της C (όχι συνάρτηση βιβλιοθήκης)
  - Επειδή μόνο η γλώσσα (compiler) γνωρίζει τα μεγέθη των τύπων
  - Για αυτό τον λόγο μπορούμε και να γράψουμε
    - s = sizeof(x); s = sizeof x; s = sizeof int; s = sizeof(int);

Βιβλιοθήκες – Αρχικά Input / Output

- Η γλώσσα C αναπαριστά όλα τα περιφερειακά σαν ένα stream από χαρακτήρες (chars)
- Κάθε πρόγραμμα όταν ξεκινάει γνωρίζει (από την γλώσσα) τρία streams που ονομάζονται:
  - stdout: οθόνη (write only stream)
  - stdin: πληκτρολόγιο (read only stream)
  - stderr: οθόνη (read only stream) -- το stream αυτό “πολυπλέκεται” στην οθόνη με το stdout
- Όταν ένα πρόγραμμα θέλει να τυπώσει κάτι στην οθόνη, γράφει τους αντίστοιχους χαρακτήρες στο stream stdout (ή stderr κατά περίπτωση)
- Όταν ένα πρόγραμμα θέλει να διαβάσει κάτι από το πληκτρολόγιο, διαβάζει χαρακτήρες από το stdin

<stdio.h>

- Για να χρησιμοποιήσουμε την βιβλιοθήκη για Input/Output (I/O) πρέπει να βάλουμε στην αρχή του προγράμματός μας:
  - #include <stdio.h>
  - Σε επόμενο μάθημα θα εξηγήσουμε σε λεπτομέρεια τι σημαίνει και τι κάνει αυτό
- Οι συναρτήσεις που μπορούμε να χρησιμοποιήσουμε είναι
  - c = fgetc(stdin) / fputc(c, stdout): διάβασε/γράψε ένα χαρακτήρα στο αντίστοιχο stream
  - c = getchar() / putchar(c): ακριβώς το ίδιο με τις παραπάνω αλλά για την getchar το stream εννοείται ότι είναι το stdin και για την putchar το stdout
- Επίσης υπάρχουν και συναρτήσεις για ”formatted input/output” για ευκολία
  - Οι γνωστές μας fprintf/fscanf κοκ
- Πως δηλώνουμε το τέλος ενός stream, ώστε ένα πρόγραμμα να καταλαβαίνει πότε τελείωσε το input και να μην περιμένει τον επόμενο χαρακτήρα για πάντα;
  - Ορίζουμε έναν νέο, ειδικό χαρακτήρα, μέρος της γλώσσας: EOF ή End-of-File
  - Όταν διαβάσουμε το EOF σημαίνει ότι το stream μας έχει τελειώσει
  - Η τιμή του ονόματος EOF είναι implementation dependent (αρκετές φορές -1) και μπορείτε να την βρείτε στο αρχείο stdio.h
  - Το EOF δεν σχετίζεται με το NULL, NUL
    - NULL είναι η τιμή ενός null pointer και είναι implementation dependent (πολλές φορές 0, αλλά όχι απαραίτητα)
    - NUL είναι η τιμή του χαρακτήρα που τερματίζει ένα string, είναι πάντα ο πρώτος χαρακτήρας στο character set και επομένως μπορεί πάντα να γράφεται και σαν ‘\0’

Interfaces Βιβλιοθηκών στη C (ANSI C)

- The C Library Reference Guide, Eric Huss, 1997
  (link in course web page under course material)

Interfaces Βιβλιοθηκών στη C (ANSI C)

- (online) The C Library Reference Guide, Eric Huss, 1997
  (link in course web page under course material)

```
2.12.5    Character I/O Functions
              2.12.5.1  fgetc
              2.12.5.2  fgets
              2.12.5.3  fputc
              2.12.5.4  fputs
              2.12.5.5  getc
              2.12.5.6  getchar
              2.12.5.7  gets
              2.12.5.8  putc
              2.12.5.9  putchar
              2.12.5.10 puts
              2.12.5.11 ungetc
```

```
2.14 string.h
              2.14.1    Variables and Definitions
              2.14.2    memchr
              2.14.3    memcmp
              2.14.4    memcpy
              2.14.5    memmove
              2.14.6    memset
              2.14.7    strcat
              2.14.8    strncat
              2.14.9    strchr
              2.14.10   strcmp
              2.14.11   strncmp
              2.14.12   strcoll
              2.14.13   strcpy
              2.14.14   strncpy
              2.14.15   strcspn
              2.14.16   strerror
              2.14.17   strlen
              2.14.18   strpbrk
              2.14.19   strrchr
              2.14.20   strspn
              2.14.21   strstr
              2.14.22   strtok
              2.14.23   strxfrm
```

Δομή ενός προγράμματος σε C

```
#include <stdlib.h>
#include <stdio.h>
int a,b; /* global variables */

void hello(void){
        int a, b; /* local vars */
        printf("hello hy255\n");
}

int main(int argc, char *argv[]){
        hello();
        return 0;
}
```

- Δηλώσεις
- Κώδικας σε συναρτήσεις
- Πάντα μια συνάρτηση main

How to compile/run a C program

- 1. Compile a program
  - `$ gcc -Wall -pedantic -ansi -o myprog myprog.c`
  - Πάντα χρησιμοποιούμε “–Wall –pedantic” για να βλέπουμε όλα τα errors/warnings
  - Για το μάθημα χρησιμοποιούμε πάντα και ”-ansi” (-std=C89) για να έχουμε ένα reference point
    - The C Library Reference Guide, Eric Huss, 1997
    - -std=C99 could be another option, but not appropriate for kernel-level programming
  - Αν δεν ορίσουμε όνομα για το εκτελέσιμο πρόγραμμα (-o myprog) το πρόγραμμα που παράγεται έχει πάντα όνομα a.out (από το ”assembler output”)
- 2. Run a program
  - $ ./a.out

Example: Αντιγραφή του stdin στο stdout χαρακτήρα-χαρακτήρα

- Source file example.c

```
#include <stdio.h>

int main(void){
    char c;
    while ((c=getchar()) != EOF){
            putchar(c);
    }
    return 0;
}
```

- Compile example.c
  - $ gcc –ansi –Wall –pedantic example.c
  - (online) The C Library Reference Guide, Eric Huss, 1997
- Run
  - $ ./a.out

Εξάσκηση ελεύθερου χρόνου

- 1. Γράψτε ένα απλό πρόγραμμα που να τυπώνει τα μεγέθη όλων των βασικών τύπων σε κάποιο σύστημα του Τμήματος που χρησιμοποιούμε στο HY255
- 2. Επαναλάβετε το ίδιο για ένα δικό σας σύστημα
- 3. Επαναλάβετε το ίδιο με μερικούς δομημένους τύπους που θα ορίσετε εσείς
- 4. Γράψτε ένα πρόγραμμα που αντιγράφει το stdin στο stdout χαρακτήρα-χαρακτήρα

Reading

- Επανάληψη
  - C reference card [pdf]
  - stdio.h in Section 2.12.{1,2,3} of The C Library Reference Guide, Eric Huss, 1997. [html]
  - King Ch.1-6, 7.{1-3}, 8, 9.1, 16, 22.{1,3,4}

  or
  - Ch. 1,2,3,6 of (online) The C Book, Mike Banahan, Declan Brady and Mark Doran, Second Edition, 2003. [html] [pdf (updated 2020)]
- Ξεφυλλίστε (στο web page του μαθήματος):
  - (online) The C Library Reference Guide, Eric Huss, 1997. [html]
  - (online) Ch. 9 of The C Book, Mike Banahan, Declan Brady and Mark Doran, Second Edition, 2003. [html] [pdf (updated 2020)]