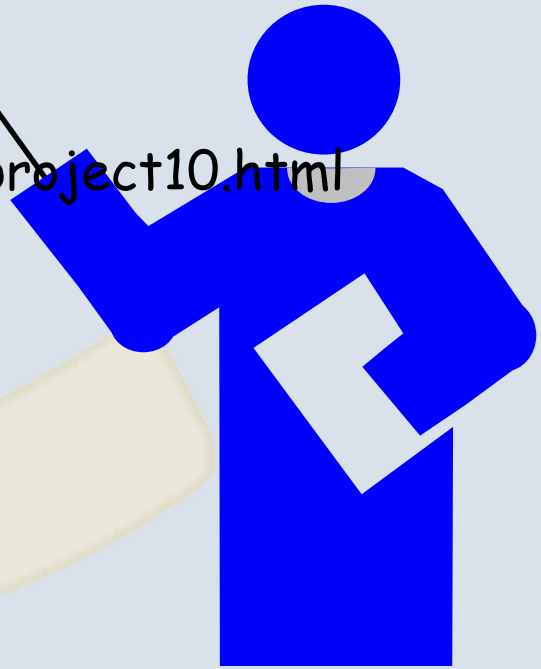
Software Project

Lecturers: Ran Canetti & Roded Sharan
Teaching Assistants: Nathan Manor & Ben Riva

Tel Aviv University
Email: (roded/canetti/benriva)@post.tau.ac.il
nathan.manor@gmail.com

http://www.cs.tau.ac.il/~roded/courses/soft-project10.html

A Book on C - ABC
Kelley/Pohl
Addison-Wesley
Fourth Edition



Course Structure

- The C programming language: 8-9 classes including 3 exercises (20% of final grade).
- Large project: in pairs (50% of final grade).

  2 classes will be devoted to project presentation.
- Final exam: 30% of final grade.

The C programming language

- B, initial versions of UNIX, 1970
- C, Dennis Ritchie, Bell Labs, 1972; overcomes B's typeless representation; used for developing UNIX.
- Early 80's - traditional C
- 1990 – the ANSI-C standard
- C++ (1980's) and Java (1990's)

The C design philosophy

- Minimal constructs and abstractions
- Keep tight control over the object code
- Allow a simple, efficient compiler
- Provide programming flexibility

Why learn C ?

•C is **powerful and efficient**: Allows direct memory management, address manipulation, direct modification of machine registers. Efficient compiler.

→A language of choice for large code that has to be optimized, systems.

• C is **compact**: terse commands, few reserved words.

• C is **modular** and **typed**.

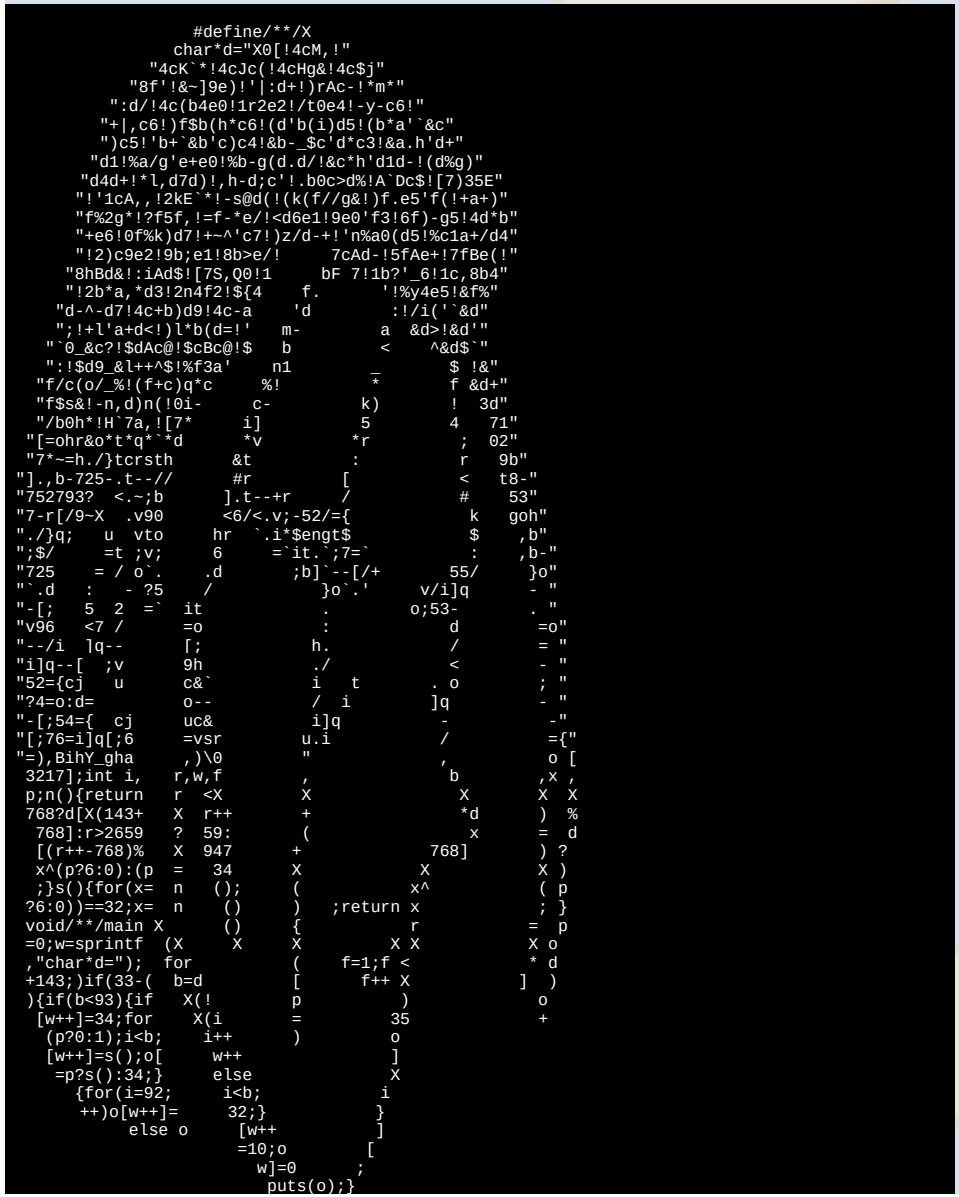
• C is the native language of **UNIX**.

• Basis of C++ (and Java).

Why learn C ?

- C is fun!
- C is also a culture…
- There are international contests in writing efficient and “creative” C code

A winning entry in the 15th Int'l Obfuscated C Code Contest (IOCCC'00)



```
               #define/**/X
              char*d="X0[!4cM,!"
            "4cK`*!4cJc(!4cHg&!4c$j"
          "8f'!&~]9e)!'|:d+!)rAc-!*m*"
        ":d/!4c(b4e0!1r2e2!/t0e4!-y-c6!"
       "+|,c6!)f$b(h*c6!(d'b(i)d5!(b*a'`&c"
       ")c5!'b+`&b'c)c4!&b-_$c'd*c3!&a.h'd+"
      "d1!%a/g'e+e0!%b-g(d.d/!&c*h'd1d-!(d%g)"
     "d4d+!*l,d7d)!,h-d;c'!.b0c>d%!A`Dc$![7)35E"
    "!'1cA,,!2kE`*!-s@d(!(k(f//g&!)f.e5'f(!+a+)"
    "f%2g*!?f5f,!=f-*e/!<d6e1!9e0'f3!6f)-g5!4d*b"
    "+e6!0f%k)d7!+~^'c7!)z/d-+!'n%a0(d5!%c1a+/d4"
    "!2)c9e2!9b;e1!8b>e/!     7cAd-!5fAe+!7fBe(!"
   "8hBd&!:iAd$![7S,Q0!1     bF 7!1b?'_6!1c,8b4"
    "!2b*a,*d3!2n4f2!${4    f.      '!%y4e5!&f%"
   "d-^-d7!4c+b)d9!4c-a    'd        :!/i(`&d"
    ";!+l'a+d<!)l*b(d=!'   m-        a  &d>!&d'"
   "`0_&c?!$dAc@!$cBc@!$   b         <    ^&d$`"
    ":!$d9_&l++^$!%f3a'    n1        _       $ !&"
   "f/c(o/_%!(f+c)q*c     %!         *       f &d+"
    "f$s&!-n,d)n(!0i-     c-         k)      !  3d"
    "/b0h*!H`7a,![7*     i]          5       4   71"
  "[=ohr&o*t*q*`*d      *v         *r         ;  02"
  "7*~=h./}tcrsth      &t          :          r   9b"
 "].,b-725-.t--//      #r         [           <   t8-"
 "752793?  <.~;b      ].t--+r     /           #    53"
 "7-r[/9~X  .v90      <6/<.v;-52/={            k   goh"
 "./}q;   u  vto     hr   `.i*$engt$           $    ,b"
 ";$/     =t ;v;     6     =`it.`;7=`          :    ,b-"
 "725    = / o`.    .d       ;b]`--[/+       55/     }o"
 "`.d   :   - ?5    /          }o`.'     v/i]q      - "
 "-[;   5  2  =`  it           .       o;53-       . "
 "v96   <7 /      =o           :          d        =o"
 "--/i  ]q--      [;           h.         /        = "
 "i]q--[  ;v      9h           ./         <        - "
 "52={cj   u      c&`          i   t      . o       ; "
 "?4=o:d=         o--          /  i       ]q        - "
 "-[;54={  cj     uc&          i]q         -         -"
 "[;76=i]q[;6     =vsr        u.i          /         ={"
 "=),BihY_gha     ,)\0        "            ,         o [
 3217];int i,   r,w,f        ,             b        ,x ,
 p;n(){return   r  <X        X              X       X  X
 768?d[X(143+   X  r++       +              *d      )  %
  768]:r>2659   ?  59:       (               x      =  d
  [(r++-768)%   X  947      +             768]      ) ?
  x^(p?6:0):(p  =   34      X           X           X )
  ;}s(){for(x=  n   ();     (           x^          ( p
 ?6:0))==32;x=  n    ()     )   ;return x           ; }
 void/**/main X      ()     {           r          =  p
 =0;w=sprintf  (X     X     X         X X          X o
 ,"char*d=");  for          (    f=1;f <           * d
 +143;)if(33-(  b=d         [      f++ X          ]  )
 ){if(b<93){if   X(!        p          )            o
  [w++]=34;for    X(i       =         35            +
   (p?0:1);i<b;    i++      )         o
   [w++]=s();o[     w++               ]
    =p?s():34;}     else              X
      {for(i=92;     i<b;            i
       ++)o[w++]=     32;}           }
            else o     [w++         ]
                       =10;o       [
                         w]=0      ;
                          puts(o);}
```

The author (D.H. Yang):
"Instead of making one self-reproducing program, what I made was a program that generates a set of mutually reproducing programs, all of them with cool layout!"

Winner of IOCCC'04

```
#define G(n) int n(int t, int q, int d) #define X(p,t,s) (p>=t&&p<(t+s)&&(p-
(t)&1023)<(s&1023)) #define U(m) *((signed char *)(m)) #define F if(!--q){ #define I(s)
(int)main-(int)s #define P(s,c,k) for(h=0; h>>14==0;
h+=129)Y(16*c+h/1024+Y(V+36))&128>>(h&7)?U(s+(h&15367))=k:k G (B) { Z; F D =
E (Y (V), C = E (Y (V), Y (t + 4) + 3, 4, 0), 2, 0); Y (t + 12) = Y (t + 20) = i; Y (t + 24) =
1; Y (t + 28) = t; Y (t + 16) = 442890; Y (t + 28) = d = E (Y (V), s = D * 8 + 1664, 1, 0);
for (p = 0; j < s; j++, p++) U (d + j) = i == D | j < p ? p--, 0 : (n = U (C + 512 + i++)) < ' '
? p |= n * 56 - 497, 0 : n; } n = Y (Y (t + 4)) & 1; F U (Y (t + 28) + 1536) |= 62 & -n; M U
(d + D) = X (D, Y (t + 12) + 26628, 412162) ? X (D, Y (t + 12) + 27653, 410112) ? 31 : 0 :
U (d + D); for (; j < 12800; j += 8) P (d + 27653 + Y (t + 12) + ' ' * (j & ~511) + j % 512,
U (Y (t + 28) + j / 8 + 64 * Y (t + 20)), 0); } F if (n) { D = Y (t + 28); if (d - 10) U (++Y (t
+ 24) + D + 1535) = d; else { for (i = D; i < D + 1600; i++) U (i) = U (i + 64); Y (t + 24) =
1; E (Y (V), i - 127, 3, 0); } } else Y (t + 20) += ((d >> 4) ^ (d >> 5)) - 3; } } G (_); G (o);
G (main) { Z, k = K; if (!t) { Y (V) = V + 208 - (I (_)); L (209, 223) L (168, 0) L (212,
244) _((int) &s, 3, 0); for (; 1;) R n = Y (V - 12); if (C & ' ') { k++; k %= 3; if (k < 2) { Y
(j) -= p; Y (j) += p += U (&D) * (1 - k * 1025); if (k) goto y; } else { for (C = V - 20; !i
&& D & 1 && n && (X (p, Y (n + 12), Y (n + 16)) ? j = n + 12, Y (C + 8) = Y (n + 8), Y
(n + 8) = Y (V - 12), Y (V - 12) = n, 0 : n); C = n, n = Y (n + 8)); i = D & 1; j &= -i; } }
else if (128 & ~D) { E (Y (n), n, 3, U (V + D % 64 + 131) ^ 32); n = Y (V - 12); y:C = 1
<< 24; M U (C + D) = 125; o (n, 0, C); P (C + p - 8196, 88, 0); M U (Y (0x11028) + D) =
U (C + D); } } } for (D = 720; D > -3888; D--) putchar (D > 0 ? "
)!\320\234\360\256\370\256 0\230F .,mnbvcxz ;lkjhgfdsa \n][poiuytrewq =-0987654321
\357\262 \337\337 \357\272 \337\337 ( )\"\343\312F\320!/ !\230 26!/\16 K>!/\16\332
\4\16\251\0160\355&\2271\20\2300\355`x{0\355\347\2560 \237qpa%\231o!\230
\337\337\337 , )\"K\240 \343\316qrpxzy\0 sRDh\16\313\212u\343\314qrzy !0( " [D] ^ 32
: Y (I (D))); return 0; } G (o) { Z; if (t) { C = Y (t + 12); j = Y (t + 16); o (Y (t + 8), 0, d);
M U (d + D) = X (D, C, j) ? X (D, C + 1025, j - 2050) ? X (D, C + 2050, j - 3075) ? X (D,
C + 2050, j - 4100) ? X (D, C + 4100, ((j & 1023) + 18424)) ? 176 : 24 : 20 : 28 : 0 : U (d
+ D); for (n = Y (t + 4); U (i + n); i++) P (d + Y (t + 12) + 5126 + i * 8, U (n + i), 31); E
(Y (t), t, 2, d); } } G (_) { Z = Y (V + 24); F Y (V - 16) += t; D = Y (V - 16) - t; } F for (i =
124; i < 135; i++) D = D << 3 | Y (t + i) & 7; } if (q > 0) { for (; n = U (D + i); i++) if (n -
U (t + i)) { D += _(D, 2, 0) + 1023 & ~511; i = ~0; } F if (Y (D)) { n = _(164, 1, 0); Y (n +
8) = Y (V - 12); Y (V - 12) = n; Y (n + 4) = i = n + 64; for (; j < 96; j++) Y (i + j) = Y (t +
j); i = D + 512; j = i + Y (i + 32); for (; Y (j + 12) != Y (i + 24); j += 40); E (Y (n) = Y (j +
16) + i, n, 1, 0); } } } return D; }
```

The author, Gavin Barraclough: "This is a 32-bit multitasking operating system for x86 computers, with GUI and filesystem, support for loading and executing user applications in elf binary format, with ps2 mouse and keyboard drivers, and vesa graphics. And a command shell. And an application - a simple text-file viewer."

C vs. Java

Java

C++

C

Assembly Language

C vs. Java

• Java was developed based on C and inherited most of its syntax.

• Main difference: C is procedural and not object-oriented. Thus, no classes/interfaces; methods are called functions and do not belong to any object.

• Similar primitive types as in Java, but no boolean and string types; other differences.

• *C is much less structured. It is easier to make mistakes and harder to recover.*

C vs. Java (cont.)

• Pointers instead of references. Allow direct access to memory and dereferencing (but less elegant).

• No garbage collection – need to manage memory explicitly.

• Preprocessor.

• Compilation to machine code – faster but some platform dependency; java is processed via an interpreter (JVM).

C vs. Java – small differences

- No for/in (iterating within a set)
- No function overloading
- Arrays are not objects – cannot apply methods to them (e.g. no cloning).
- Global variables.
- Variable declarations only in the beginning of a block.
- Composite types such as union, bitfield; typedef.
- Function pointers.

C Topics

- Lexical elements (Ch. 2)
- Fundamental data types (Ch. 3)
- Flow of control (Ch. 4)
- Functions (Ch. 5); Runtime environment
- Arrays, pointers and strings (Ch. 6); Dynamic matrix allocation (Ch. 12.6)
- Bitwise operators (Ch. 7)
- Preprocessor (Ch. 8)
- Structures and enumeration types (Chs. 9 & 7.5)
- Linked lists (Ch. 10)
- Input/output and files (Ch. 11)

Programming Environment

- Unix
- Make

Operating System

Operating system:

- Manages the available hardware (CPU, memory…) and software (editors, development tools…) resources.
- Provides interface for using them by multiple users.

Users → AUI → API → OS kernel → Hardware

Shell: command-line interpreter to interface the OS.

The Unix OS

Unix: multi-user, multi-process OS; open source.

History:

- 1969 – Initial version by Thompson & Ritchie at Bell Labs (assembly and later B).
- 70's – Rewritten in C.
- 80's – System-V (AT&T) and BSD (Berkeley) versions.
- 90's – Linux by Linus Torvalds.

For basic introduction and commands see web-page.

The Programming Project

Goals:

Build software system.

Memory management.

Fit spec & understand complex requirements.

Basic file & disc handling.

Receive & interface with external code.

Testing and debug purpose functions.

Missions story - Security

WindowsME had a serious weakness in password authentication method - trust “one-way” function that is actually easy to invert.

http://www.ethicalhacker.net/content/view/94/24

http://en.wikipedia.org/wiki/LM_hash

Characters	A-Z,0-9,!-=
Number of possible passwords	~2^40
Disc-space	24 GB

Breaking using rainbow-table

- Preprocessing all possible passwords, and save it efficiently.
- Disc-space vs. Access-time tradeoff.

Assignments

Ex1 (5%) : Break RSA on 32bit keys.

Flow-control, smart use of variables.

Ex2 (5%) : Legal password generation.

String manipulation, simplest memory management.

Ex3(10%): Implementing hash-table

Lists, correct memory management.

Project (50%): Passwords authentication, and its retrieving using rainbow tables.

Large system, accessing to disc, files.

Huge runs (debug with small runs).

Advise from experience

Porting – don’t postpone to last 48 hours.

- **Unix is more stringent than Windows.**
- **Basement is crowded during money time.**
- **Unexpected surprises…**

Testing. Think a lot about testing.

- **OK to copy test files from friends, if not submitted.**
  - **You cannot expect being bit-exact with others.**
- **Both small scale tests and large scale.**

We will not provide 100 private lessons.

Assistants perspective

Ben & Nathan

- **We bring extensive experience in software engineering.**
- **Exercises & project definition, explanations**
  - **This year new team, new project.**
- **Supporting forums for your questions**
  - **Team email for personal issues.**
- **Frontal teaching regarding project**
  - **Unix environment, Makefile, etc.**

From source code to executable

Three main steps:

- **Preprocessing:** First pass on the file that substitutes human-friendly text with machine-firendly representation
- **Compilation:** Generation of object code
- **Linking:** Combining several object code files into a single executable set of instructions.

Example 1

```
#include <stdio.h>
int main(void) {
 printf("Hello, world!\n");
 return 0;
}
```

Source -> preprocssing -> compiling -> linking -> executable

hello.c ***"gcc hello.c -o hello"*** ***hello***

Example 2

```
#include <stdio.h>
int main(void) {
   int c;
   while ((c = getchar()) != EOF)
        putchar(c);
   return 0;
 }
```

Example 3

```
#include <stdio.h>
void primes(int cap) {
    int i, j, composite;
    for(i = 2; i < cap; ++i) {
      composite = 0;
      for(j = 2; j * j <= i; ++j)
        composite += !(i % j);
      if(!composite)
        printf("%d\t", i);
     }
  }
  int main() {
    primes(100);
  }
```

Example 4: What does this one do?

```
#include <stdio.h>
 _(__,___,____,_____){___/__<=_____?_(__,___+_____,____,_____):
!(___%__)?_(__,___+_____,___%__,_____):___%__==___/__&&!___
_?(printf("%d\t",___/__),_(__,___+_____,____,_____)):___%__>1&
&___%__<___/__?_(__,_____+___,____+!(___/__%(___%__),_____))
:___<__*__?_(__,___+_____,____,_____):0;}main(){_(100,0,0,1);}
```

Example 5: bubble sort

(bubble_sort.c)

```
#include <stdio.h>
#define CLASS_SIZE 5
int main(void) {
  int i, j, score[CLASS_SIZE], sum = 0, tmp;
  printf("Input %d scores: ", CLASS_SIZE);
  for (i = 0; i < CLASS_SIZE; ++i) {
     scanf("%d", &score[i]);
     sum += score[i];
  }
  for (i = 0; i < CLASS_SIZE - 1; ++i) /* bubble sort */
     for (j = CLASS_SIZE - 1; j > i; --j) if (score[j-1] < score[j]) {/* check the order */
        tmp = score[j-1]; score[j-1] = score[j]; score[j] = tmp;
  }
  printf("\nOrdered scores:\n\n");
  for (i = 0; i < CLASS_SIZE; ++i)
              printf(" score[%d] =%5d\n", i, score[i]);
  printf("\n%18d%s\n%18.1f%s\n\n",
           sum, " is the sum of all the scores", (double) sum / CLASS_SIZE, " is the class average");
  return 0; }
```

Example 5: bubble sort

output:

Input 5 scores: 63 88 97 53 77

Ordered scores:

score[0] = 97

score[1] = 88

score[2] = 77

score[3] = 63

score[4] = 53

378 is the sum of all the scores

75.6 is the class average