Pointer

Pointers I - introducing

Recap

Memory - In computing, memory refers to the computer hardware devices used to store information for immediate use in a computer; https://en.wikipedia.org/wiki/Computer_memory

Variable - a symbolic name associated with a value and whose associated value may be changed.

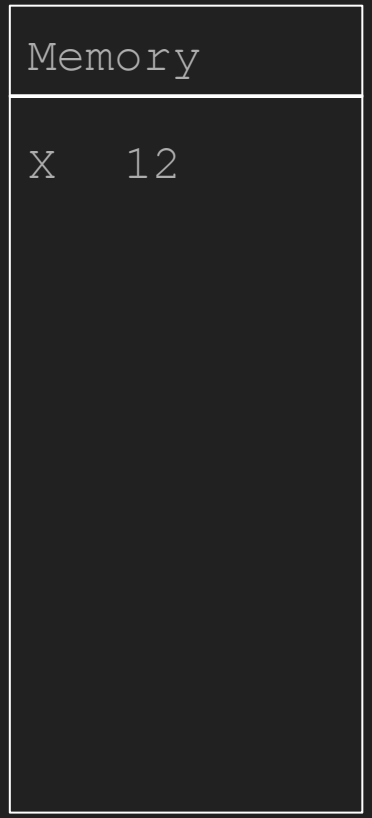
Variable

```
int x;

x = 12;
```

Variable

```
int x;

x = 12;
```

Direct!



Pointers I

Can you paint my apartment?

Yes, can you bring your house to me?

What?

Ah, I see. Can you give me the address?

House painter by nicubunu - openclipart.org

Real life

Mystreet 65
Dream Town
Imaginationland

Real life

Mystreet 65
Dream Town
Imaginationland

Pointers I

Gee whiz!

My work here is done!

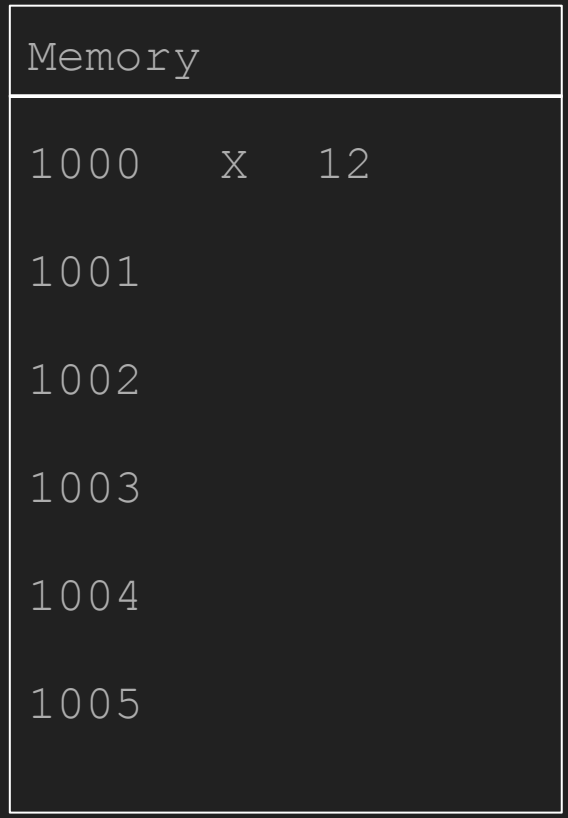
Real life

Mystreet 65
Dream Town
Imaginationland

Pointers II - address of a variable

Variable

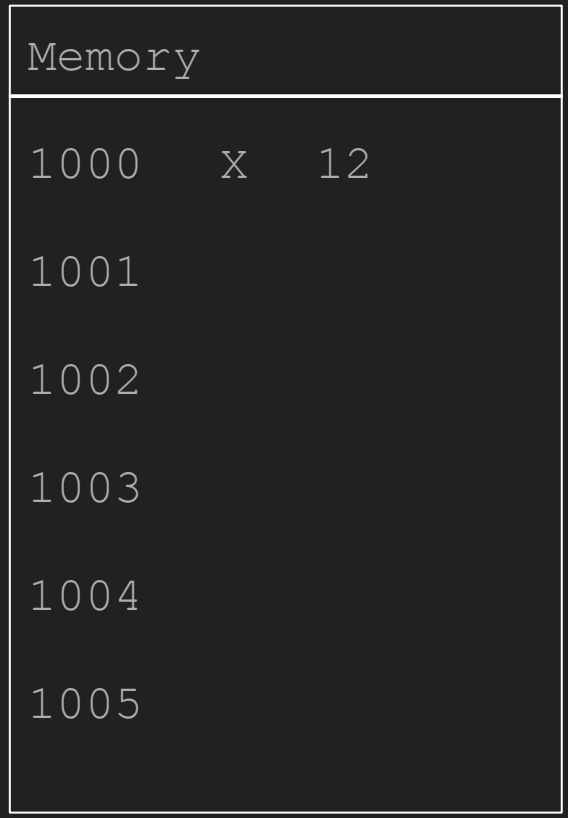
```
int x=12;
```



Variable

```
int x=12;
```

What’s the address of x?

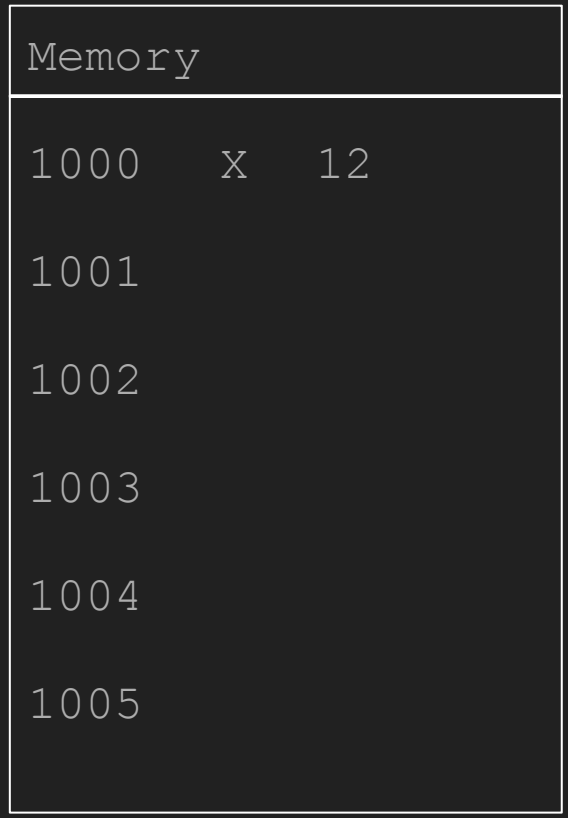


Variable

```
int x=12;
```

What's the address of x?

It's `1000`



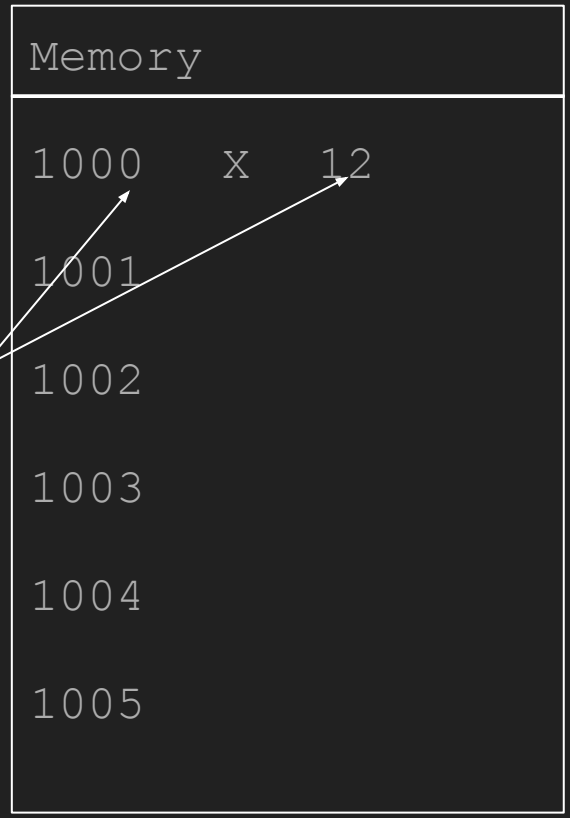
& - address of operator

Gimme me the address of ..

```c
#include <stdio.h>


int main()
{
  int x = 12;
  printf ("value:   %d \n", x);
  printf ("address: %p \n", &x);
}
```

Variable

```c
#include <stdio.h>

int main()

{

  int x = 12;

  printf ("value:   %d \n", x);

  printf ("address: %p \n", &x);

}
```



// https://github.com/progund/programming-with-c/blob/master/c-pointers/presentation-source/pointer-example.c

```
# gcc pointer-example.c -o pointer-example && ./pointer-example

value:   12

address: 0x7fffba538a14
```

For the fun of it

Download one the command line:

- `wget "https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example.c`
- `curl "https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example.c" -o pointer-example.c`

Download and compile:

- `wget "https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example.c" && gcc pointer-example.c -o pointer-example && ./pointer-example`
- `curl "https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example.c" -o pointer-example.c && gcc pointer-example.c -o pointer-example && ./pointer-example`

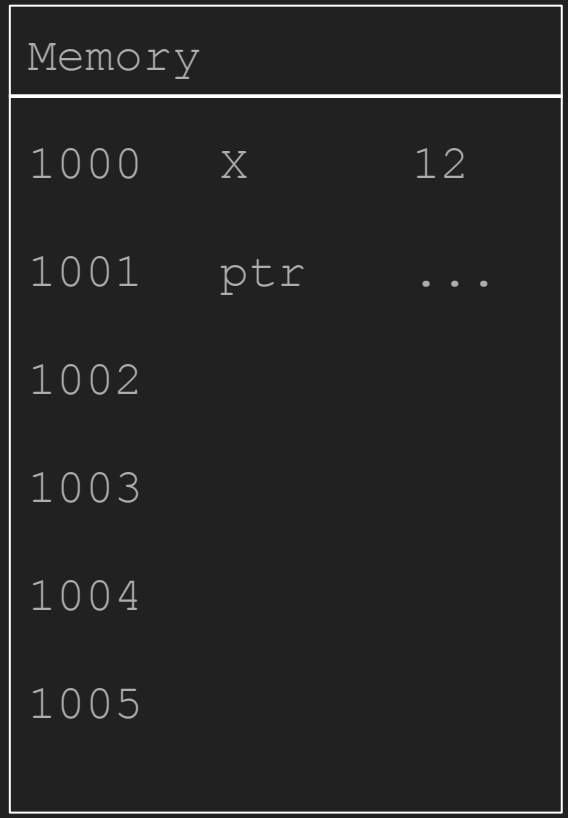
Pointers III - declaring a pointer

Declaring a pointer to int

```
int *ptr;
```

Variable

```
int x = 12;

int *ptr;
```



*

int *iptr;	iptr is a pointer that points to int
double *dptr;	dptr is a pointer that points to double
media *mp;	mp is a pointer that points to a media

Using pointers to store an address

int c=12;	Declare an int var, called c. Assign it 12
int *cp = &c;	Declare a pointer to int variable, called cp. Assign it the address of c

Variable

```
#include <stdio.h>

int main()

{

  int x   = 12;

  int *xp = &x;

  printf ("value:   %d \n", x);

  printf ("address: %p \n", &x);

  printf ("address: %p \n", xp);

}

// https://github.com/progund/programming-with-c/blob/master/c-pointers/presentation-source/pointer-example2.c
```

```
# gcc pointer-example2.c -o pointer-example2 && ./pointer-example2

value:   12

address: 0x7ffe9ed38c8c

address: 0x7ffe9ed38c8c
```

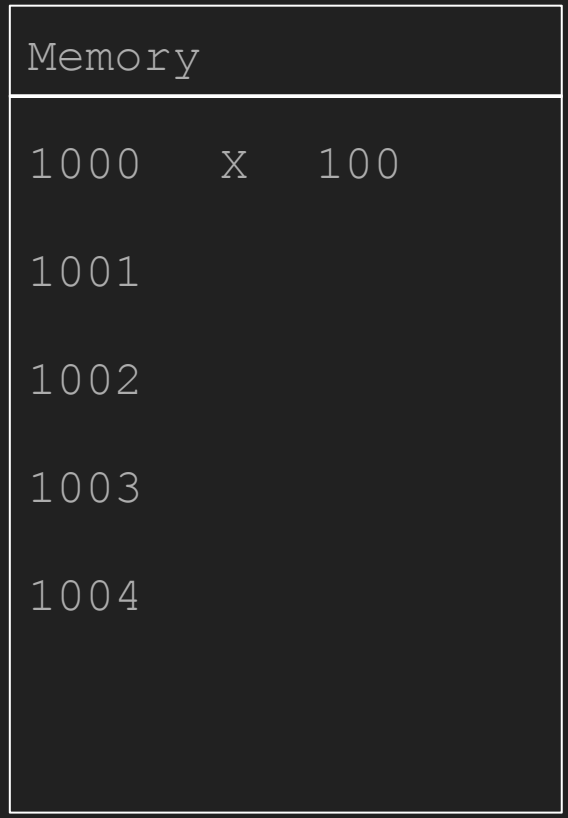
Pointers IV - dereferencing a pointer

Let’s add some more to our super mega program.

This time we’re going to use * in another way

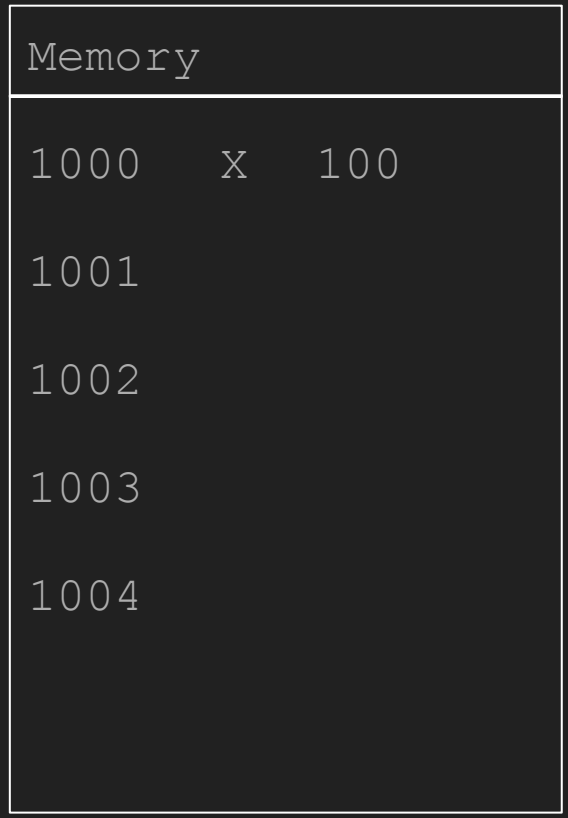
* again

```
int x = 100;

int *xp = &x;

int y = *xp;
```



* again

```
int x = 100;

int *xp = &x;

int y = *xp;
```



* again

```
int x = 100;

int *xp = &x;

int y = *xp;
```

Memory		
1000	X	100
1001		
1002	xp	1000
1003		
1004		
1005		

Some theory

*xp ; Dereference operator

“Go to the address of xp”

* again

```
int x = 100;

int *xp = &x;

int y = *xp;
```

Memory		
1000	X	100
1001		
1002	xp	1000
1003		
1004	y	100
1005		

```c
#include <stdio.h>

int main()

{

  int  x  = 12;

  int *xp = &x;

  int  y  = *xp;

  printf ("x value:   %d \n", x);

  printf ("x address: %p \n", &x);

  printf ("xp:        %p \n", xp);

  printf ("y value:   %d \n", y);

  printf ("y address: %p \n", &y);

}

// https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example3.c
```

```
# gcc pointer-example3.c -o pointer-example3 && ./pointer-example3
x value:   12
x address: 0x7ffeabdbf7f8
xp:        0x7ffeabdbf7f8
y value:   12
y address: 0x7ffeabdbf7fc
```

Let’s add some more

```
int x = 100;

int *xp = &x;

int y = *xp;

*xp = y * 2; ← added
```

Memory		
1000	X	100
1001		
1002	xp	1000
1003		
1004	y	100
1005		

Let’s add some more

```
int x = 100;

int *xp = &x;

int y = *xp;

*xp = y * 2;
```

Memory		
1000	X	200
1001		
1002	xp	1000
1003		
1004	y	100
1005		

```
 … add to main

  printf ("\n -- extended -- \n");

  printf ("Changing the value of x using xp (which points to x)");

  // dereference xp (which points to x)

  // assign it 24 (y * 2 => 12*2 => 24)

  *xp = y * 2;

  printf ("x value:   %d \n", x);

  printf ("x address: %p \n", &x);

  printf ("xp:        %p \n", xp);

  printf ("y value:   %d \n", y);

  printf ("y address: %p \n", &y);

}

// https://raw.githubusercontent.com/progund/programming-with-c/master/c-pointers/presentation-source/pointer-example3.c
```

```
# gcc pointer-example3.c -o pointer-example3 && ./pointer-example3

x value:   12

x address: 0x7ffe831c14c8

xp:        0x7ffe831c14c8

y value:   12

y address: 0x7ffe831c14cc



 -- extended --

Changing the value of x using xp (which points to x)x value:   24

x address: 0x7ffe831c14c8

xp:        0x7ffe831c14c8

y value:   12

y address: 0x7ffe831c14cc
```

Pointers V - pointers and functions

```c
#include <stdio.h>

int doubler(int x) {
  return x * 2;
}

int main() {
  int orig = 12;
  printf ("orig: %d\n", orig);
  orig  = doubler(12);
  printf ("orig: %d\n", orig);
  return 0;
}
```

Memory

address	name	value
1000		
1001		
1002		

```c
#include <stdio.h>

int doubler(int x) {
  return x * 2;
}

int main() {
  int orig = 12;
  printf ("orig: %d\n", orig);
  orig  = doubler(12);
  printf ("orig: %d\n", orig);
  return 0;
}
```

Memory

address	name	value
1000	orig	12
1001		
1002		

```c
#include <stdio.h>

int doubler(int x) {
  return x * 2;
}

int main() {
  int orig = 12;
  printf ("orig: %d\n", orig);
  orig  = doubler(12);
  printf ("orig: %d\n", orig);
  return 0;
}
```

Memory

address	name	value
1000	orig	12
1001		
1002		

```c
#include <stdio.h>

int doubler(int x) {

  return x * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  orig  = doubler(12);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	12
1001	x	12
1002		

```c
#include <stdio.h>

int doubler(int x) {
  return x * 2;
}

int main() {
  int orig = 12;
  printf ("orig: %d\n", orig);
  orig  = doubler(12);
  printf ("orig: %d\n", orig);
  return 0;
}
```

Memory

address	name	value
1000	orig	24
1001		
1002		

… imagine we want to update orig
in the function instead

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000		
1001		
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	12
1001		
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	12
1001		
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	12
1001	xp	1000
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	24
1001	xp	1000
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	24
1001		
1002		

```c
#include <stdio.h>

void doubler(int *xp) {

  *xp = *xp * 2;

}

int main() {

  int orig = 12;

  printf ("orig: %d\n", orig);

  doubler(&orig);

  printf ("orig: %d\n", orig);

  return 0;

}
```

Memory

address	name	value
1000	orig	24
1001		
1002		

Pointers VI - user arguments

Supplying arguments to a function

```
add(12,23);
```

How to supply arguments to main?

Pointers VI - user arguments

You already have done that

```
gcc main.c awesomemath.c unawesome.c
```

How does gcc use that? …

Pointers VI - user arguments

```
int main(int argc) {

    printf (“Nr of arguments: %d\n”);

}
```

Pointers VI - user arguments

```
int main(int argc, char **argv) {

        printf (“Nr of arguments: %d\n”);

}
```

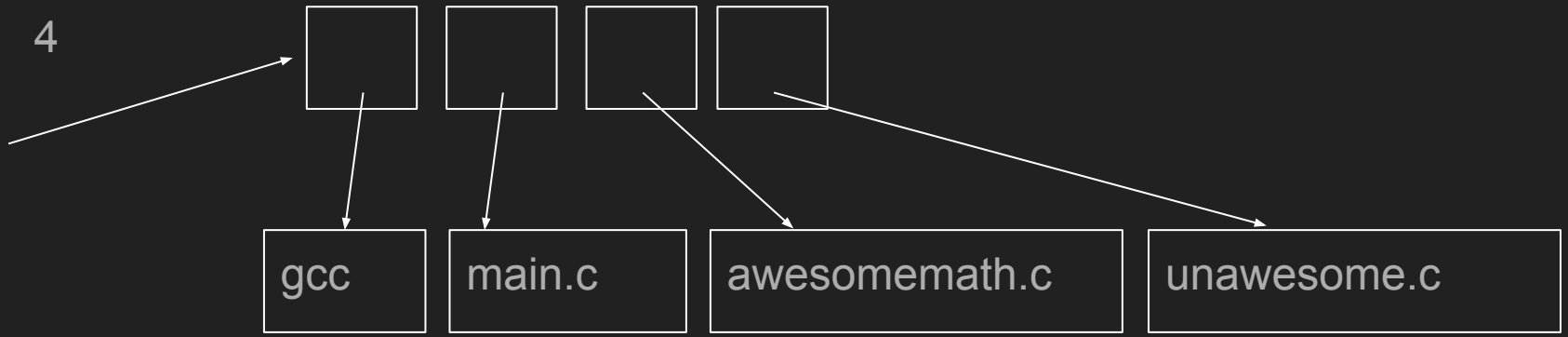
Pointers VI - user arguments

gcc main.c awesomemath.c unawesome.c

Memory

argc: 4

argv



Listing user arguments

```
int main(int argc, char **argv)

{

  int i;

  printf ("Listing user args:\n");

  printf ("-------------------\n");

  for (i=0;i<argc;i++)

  {

    printf (" * args[%d]: %s\n", i, argv[i]);

  }

}
```

Parsing user arguments

```
int

parse_args(int argc, char **argv)

{

  int i ;

  for (i=1;i<argc;i++)   // We start from 1, since 0 is the program itself

  {

    // If use supplied "--list", it shall be recognised as a valid option

    if (strncmp(argv[i], "--list", strlen("--list"))==0)

      {

        printf (" * Using long listing\n");

      }

    else
```

Parsing user arguments

```
else
      { // all other options are invalid
        fprintf(stderr, " * unknown option: '%s'\n", argv[i]);
        return SYNTAX_ERROR;
      }
  }
}
```