

[C No Evil]

A practioner's guide

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

1

[Playing with fire]

- Program arguments
- Pointer arithmetic
- Output

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

2

[ARGCount ARGValues]

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

- **argc**
 - Argument count
 - The number of arguments that are passed to **main** in the argument vector **argv**.
 - the value of **argc** is always one greater than the number of command-line arguments that the user enters.

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

3

[ARGCount ARGValues]

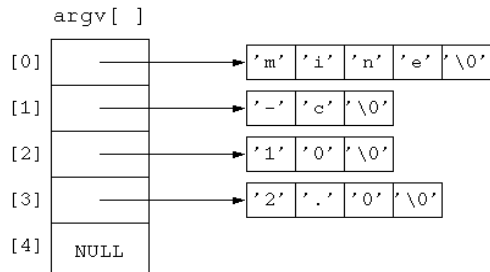
```
int main(argc, char** argv)
int main(argc, char* argv[])
```

- **argv**
 - argument vector
 - An array of string pointers passed to a C program's **main** function
 - **argv[0]** is always the name of the command
 - **argv[argc]** is a null pointer

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

4

ARGCount ARGValues



Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

5



ARGCount ARGValues

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

- `*(argv + argc)` is `NULL`
- `argv[argc]` is `NULL`

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

6



Followup: Can add integers to pointers

- Compiler uses the type information
 - `long *p;`
 - `p` ▶ `[long][long][long]`
- What address is `p + 2`?
 - ... `p + sizeof(long) * 2`

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

7



Followup: output

- C stdio library functions

```
printf("Hello %x %s %d", arguments...)
fprintf(STDOUT, STDERR, "%x%s%d", ...)
```

- Later... system call
`write(int, void*, size_t)`

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

8



printf Format Identifiers

<code>%d %i</code>	Decimal signed integer.
<code>%o</code>	Octal integer.
<code>%x %X</code>	Hex integer.
<code>%u</code>	Unsigned integer.
<code>%c</code>	Character.
<code>%s</code>	String.
<code>%f</code>	Double.
<code>%p</code>	Pointer.

All of the parameters should be the value to be inserted.
EXCEPT %s, this expects a pointer to be passed

Copyright ©: Nahstedt, Angrave, Abdelzاهر, Kravets, Gupta

9



printf Basic Data Types

```
#include <stdio.h> // for printf
int main(int argc, char *argv[]) {
    // print "the date is: 05.01.2006",
    // i.e. 2- or 4-digit with leading zeros
    // using 32-bit 'long' datatype
    long lday = 5;
    long lmonth = 1;
    long lyear = 2006;
    printf("the date is: %021d.%021d.%041d\n", lday, lmonth, lyear);

    // - print 8-digit hex value
    // - print a pointer value
    unsigned long ulID = 0x12345678;
    unsigned long *pID = &ulID;
    printf("hex value: 0x%021X at address: %p\n", ulID, pID);
}
```



printf Basic Data Types

```
// - print 4 bytes of a 32-bit ulong value
// as separate hex values
unsigned char uc1 = (unsigned char) (ulID >> 24);
unsigned char uc2 = (unsigned char) (ulID >> 16);
unsigned char uc3 = (unsigned char) (ulID >> 8);
unsigned char uc4 = (unsigned char) (ulID >> 0);
printf("hex bytes: %02X %02X %02X %02X\n", uc1, uc2, uc3, uc4);

// - print double value like "70.35000"
double dTemp = 70.35;
printf("temperature: %5.5f\n", dTemp);
}
```

printf Escape Sequences

<code>\a</code>	<bell>	<code>\'</code>	<single quote>
<code>\b</code>	<backspace>	<code>\"</code>	<double quote>
<code>\e</code>	<escape>	<code>\?</code>	<question mark>
<code>\f</code>	<form-feed>	<code>\\</code>	<backslash>
<code>\n</code>	<new-line>	<code>\num</code>	an 8-bit character with ASCII value of the 1-, 2-, or 3-digit octal number <i>num</i> .
<code>\r</code>	<carriage return>		
<code>\t</code>	<tab>		
<code>\v</code>	<vertical tab>	BUT	
<code>\0</code>	<null>	<code>%%</code>	<percent>



Copyright ©: Nahstedt, Angrave, Abdelzاهر, Kravets, Gupta

12



[Followup: Typecasting]

- C allows programmers to perform typecasting by
 - Place the type name in parentheses and place this in front of the value

```
main() {  
    float a;  
    a = (float)5 / 3;  
}
```

- Result is 1.666666
 - Integer 5 is converted to floating point value before division and the operation between float and integer results in float

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

13



[Typecasting]

```
main() {  
    int a = 5000, b = 7000 ;  
    long int c = a * b ;  
}
```

- Two integers are multiplied
 - Result is truncated and stored in variable `c` of type `long int`.
 - Incorrect answer

```
long int c = (long int) a * b;
```

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

14



[Typecasting]

- Take care about using typecast
- If used incorrectly, may result in loss of data
 - Truncating a `float` when typecasting to an `int`

Copyright ©: Nahstedt, Angrave, Abdelzaher, Kravets, Gupta

15

