

C Overview

Fall 2014

Jinkyu Jeong
(jinkyu@skku.edu)

- # for preprocessor
- Indicates where to look for printf() function
- .h file is a header file

```
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

- Entry point
- Called on program start
- Only one main() in any program

- Belongs to stdio.h
- “Hello....” is a parameter to printf()

Marathon Distance Program

- **Convert the distance to kilometers**

- 1 mile = 1.609 km = 1760 yards
- We know that the marathon length is 26 miles and 385 yards, then what is it in kilometers?
 - the answer is 42.185968

Variables, Expression and Assignment

```
/* The distance of a marathon in kilometers. */

#include <stdio.h>

int main(void)
{
    int      miles, yards;
    float   kilometers;

    miles = 26;           Assignment
    yards = 385;          expression
    kilometers = 1.609 * (miles + yards / 1760.0);
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
    return 0;
}
```

Declaration of
variables

Assignment

expression

Preprocessor

- Performs before compilation
- # indicates that this line is a directive
- **#define** for symbolic constants

```
#define PI     3.141592
#define YARDS_PER_MILE      1760
```

- **#include <file-name>** imports a header file from some where
- **#include “file-name”** from your directory

Preprocessor Example

```
#include <stdio.h>
```

```
#define AREA 2337
#define SQ_MILES_PER_SQ_KILOMETER 0.3861021585424458
#define SQ_FEET_PER_SQ_MILE (5280 * 5280)
#define SQ_INCHES_PER_SQ_FOOT 144
#define ACRES_PER_SQ_MILE 640
```

pacific_sea.h

```
/* Measuring the Pacific Sea. */
```

```
#include "pacific_sea.h"

int main(void)
{
    const int pacific_sea = AREA;      /* in sq kilometers */
    double acres, sq_miles, sq_feet, sq_inches;

    printf("\nThe Pacific Sea covers an area");
    printf(" of %d square kilometers.\n", pacific_sea);
    sq_miles = SQ_MILES_PER_SQ_KILOMETER * pacific_sea;
```

pacific_sea.c

I/O Using stdio.h

- **printf("any string or characters %d %f", a, b);**
 - An output function
 - “ ” indicates a format to be displayed
 - % is followed by a single character for a format
 - c (char), d (decimal), e (exponential), f(floating), s (string)
 - Escape with \
 - \n, \t, \”, \\
- **scanf("%d", &age);**
 - An input function
 - Takes something from the standard input, and interpret as a decimal

I/O Example

```
#include <stdio.h>

int main(void)
{
    char      c1, c2, c3;
    int       i;
    float     x;
    double   y;

    printf("\n %s \n %s", "Input three characters,",
           "an int, a float, and a double: ");

    scanf("%c %c %c %d %f %lf", &c1, &c2, &c3, &i, &x, &y);

    printf("\nHere is the data that you typed in:\n");
    printf("%3c%3c%3c%5d%17e%17e\n\n", c1, c2, c3, i, x, y);
    return 0;
}
```

Control Flow

- Each statement is executed one by one sequentially

```
statement;  
statement;  
...
```

- Special statements change the flow

- **if** (expr) a single statement OR { statements }
- **while** (expr) a single statement OR { statements }
- **for** (expr1; expr2; expr3) a single statement OR { statements }

```
expr1;  
while (expr2) {  
    statements  
    expr3;  
}
```

Control Flow Example

```
#include <stdio.h>

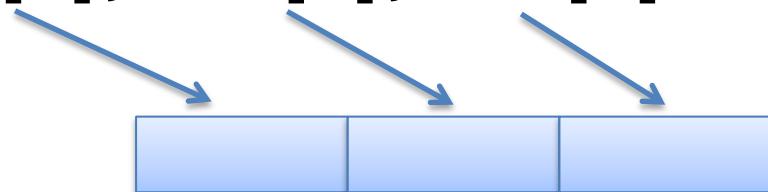
int main(void)
{
    int    i = 1, sum = 0;

    while (i <= 5) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    return 0;
}
```

Arrays

- Deal with multiple same type data
- **int xxx[3];**
 - Declares 3 integers; xxx[0], xxx[1], xxx[2]

```
int i;  
i = 2;  
xxx[i] = xxx[0] + 79;
```



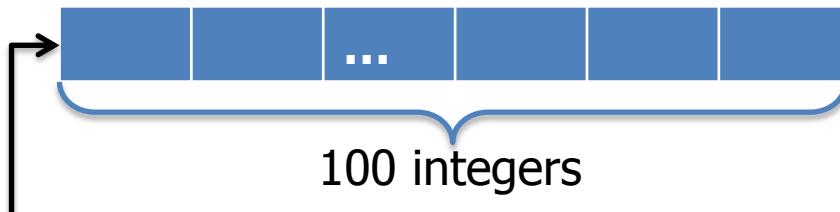
- A string “abc”



Pointer

- Address is a location in the imaginary space

- `int age[100];`



- Pointer is a special variable storing an address (location)

- Declaration of a variable with *
- `char *p;`
- `int *a = age;`



Functions

- **Can you write a program of 10,000 lines in a single file?**
 - Divide your whole code into many small chunks
 - Some chunks may look similar
 - Make them into a single one; how?
 - This is a function
 - In math
 - $y=f(x)$
 - $z=f(x, y)$
- **main() is a special function called by**

```
#include <stdio.h>

float maximum(float x, float y);
float minimum(float x, float y);
void prn_info(void);

int main(void)
{
    int i, n;
    float max, min, x;

    prn_info();
    printf("Input n: ");
    scanf("%d", &n);
    printf("\nInput %d numbers:", n);
    scanf("%f", &x);
    max = min = x;
    for (i = 2; i <= n; ++i) {
        scanf("%f", &x);
        max = maximum(max, x);
        min = minimum(min, x);
    }
    return 0;
}
```

```
float maximum(float x, float y)
{
    if (x > y)
        return x;
    else
        return y;
}

float minimum(float x, float y)
{
    if (x < y)
        return x;
    else
        return y;
}

void prn_info(void)
{
    printf("\n%s\n%s\n",
           "This program reads an",
           "integer value for n, and then",
           "processes n real numbers",
           "to find max and min values.");
}
```

Files

- You need files, believe me.
- `xfp = fopen("file-name", "r");`
 - Checks if the file is available
 - Prepares a pointer, xfp, to a location inside a file
- Now read from (write to) the file using the pointer

```
c = getc(xfp);  
n = fscanf, "%d %d %f %s", i, j, x, name);
```

```
/* Count uppercase letters in a file. */

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int      c, i, letter[26];
    FILE    *ifp, *ofp;

    ifp = fopen(argv[1], "r");
    ofp = fopen(argv[2], "w");
    for (i = 0; i < 26; ++i)          /* initialize array to zero */
        letter[i] = 0;
    while ((c = getc(ifp)) != EOF)
        if (c >= 'A' && c <= 'Z')    /* find uppercase letters */
            ++letter[c - 'A'];
    for (i = 0; i < 26; ++i) {           /* print results */
        if (i % 6 == 0)
            putc('\n', ofp);
        fprintf(ofp, "%c:%5d      ", 'A' + i, letter[i]);
    }
    putc('\n', ofp);
    return 0;
}
```

Lexical Elements, Operators, and the C System

Syntax and Compiler

```
#include <stdio.h>

int main(void)
{
    int i = 1, sum = 0;

    while (i <= 5) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    return 0;
}
```



C compiler



a.out

- The compiler understands some words and a simple grammar
- Words
 - 32 keywords: if, int, return,
 - Identifier: i, sum
 - Constants: 1, 0, 5
 - String: "sum = %d\n"
 - Operators: =, <=, +=, ++
- Grammar
 - Keyword identifier ;
 - while (expression) statement | { statements }

Compiler

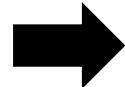
■ Task

- Generate a binary file from a source file
- 1. Check if all the words are correct (**lexical analyzer**)
- 2. Check if the grammar is correct (**syntax analyzer**)
- 3. Optimization
- 4. Code generation

Lexical Analysis

- Lexical parsing in natural language
 - 아버지 가방에 들어가신다. vs. 아버지가 방에 들어가신다.
- C follows similar rule
 - Blank (including newline) separates tokens (words)
- Lexical analysis in C
 - C program → a set of tokens (words)
 - Tokens: keywords, identifiers, constants, string constants, operators, ...

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



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

Comments

- /* anything you can put here */
- // anything you can put here (newline)
- Not a token, compiler ignores comments
- For documentation aid

```
/* The distance of
   a marathon in kilometers. */
// The distance of a marathon in kilometers.
#include <stdio.h>

int main(void)
{
    int      miles, yards;
...
```

Keywords

■ Reserved Words

- The part of C language
- Their meaning and usage predetermined

Keywords				
auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

Identifiers

- A token composed of a sequence of letters, digits and _ (underscore)
 - <identifier> ::=
 - <letter>
 - | <identifier> <letter>
 - | <identifier> <digit>
 - Underscore "_" is a letter in C
- Identifies something (variable, function, ...)
- Some identifiers are used in C library
 - printf, scanf, fopen, sqrt, pow, exp, sin, ...
 - _print

Constants

■ Integer

- 17 is a decimal
- `017` is an octal
- `0x1F` is a hexadecimal

■ Floating point

- `1.0`
- `3.14`

■ Character `'a'`, `'7'`, `','`, `'+'`, `'\n'`

String Constant

- “any thing you can put here”
- Special character should be preceded by a backslash
- It's an array of characters
- Library functions for strings
 - `strcat()`, `strcmp()`, `strcpy()`, `strlen()`

Operators

- Although C has very few keywords, it has a large number of operators

```
( ) [ ] . ->  
! ~ ++ -- + - & (type) sizeof (unary)  
* / % (binary)  
+ - (binary)  
<< >>  
< <= > >=  
== !=  
& ^ |  
&& ||  
k = (n > 0) ? a : b;  
= += ....
```

Order of evaluation
(Precedence)



Precedence and Associativity

■ Precedence

- Evaluation order at different levels of precedence

■ Associativity

- Evaluation order at the same level of precedence

■ Evaluation order examples

```
1 + 2 - 3 + 4 - 5 /* left to right */
1 + 2 * 3
(1 + 2) * 3
j = k = m = 5      /* right to left */
- - ++i            /* right to left */
i++--              /* left to right */
a * - b - c       /* (a * (-b)) - c */
```

Use parenthesis () when get confused

Increment and Decrement Operators

- Increases or decreases **a variable**
- **i++ and ++i are different**
 - **i++:** evaluate the statement and increase it
 - **++i:** increase it first, then evaluate the statement
- **Exercises**

```
int a = 1, b = 2, c = 3, d = 4;

a * b / c
a * b % c + 1
++a * b - c--
7 - -b * ++d
a = a + b++; /* after this statement, a=? b=? */
c = c + ++d; /* after this statement, c=? d=? */
```

Assignment Operators

- **= is an operator (lvalue = rvalue)**

- `a = b + c;`
- `a += b; /* equals a = a + b */`
- Same rule applies to other binary operators
 - Binary operators: `+, -, *, /, %, |, &`

- **Strange assignments**

```
a = (b = 2) + (c = 3);
a = b = c = 0; /* equals a = (b = (c = 0));
```



```
j *= k + 3;      /* j = j*(k+3) OR j = j*k +3 */
```



```
j *= k = m + 5;
```

Example

```
/* Some powers of 2 are printed. */

#include <stdio.h>

int main(void)
{
    int i = 0, power = 1;

    while (++i <= 10)
        printf("%6d", power *= 2);
    printf("\n");
    return 0;
}
```

C System

- C language, preprocessor, compiler, library, and tools

```
...  
int printf(char* fmt, ...);  
...
```

Header file (stdio.h)



```
#include <stdio.h>  
  
void main() {  
    printf("hello world");  
    return;  
}
```

Source code (hello.c)

Preprocessing

```
01.ad5d0 003c 0000 0000 0000 0000 0000 0000  
01.ad5e0 0000 0000 03f2 0000 0001 0000 0000  
01.ad5f0 0000 0000 c6c0 001a 003b 0000 0000  
01.ad600 0000 0000 0020 0000 0000 0000 0000  
01.ad610 0001 0000 0000 0000 0000 0000 0000  
01.ad620 003b 0000 0000 0000 0000 0000 0000  
01.ad630 0000 0000 0430 0000 0001 0000 0000  
01.ad640 0000 0000 c740 001a 0030 0000 0000  
01.ad650 0000 0000 0020 0000 0000 0000 0000  
01.ad660 0001 0000 0000 0000 0000 0000 0000  
01.ad670 0030 0000 0000 0000 0000 0000 0000  
01.ad680 0000 0000 045a 0000 0001 0000 0000  
01.ad690 0000 0000 c7c0 001a 0031 0000 0000  
01.ad6a0 0000 0000 0020 0000 0000 0000 0000  
01.ad6b0 0001 0000 0000 0000 0000 0000 0000  
01.ad6c0 0014 0000 0000 0000 0000 0000 0000  
01.ad6d0 0000 0000 0001 0000 0003 0000 0000  
01.ad6e0 0000 0000 c805 001a 047f 0000 0000  
01.ad6f0 0000 0001 0000 0000 0000 0000 0000
```

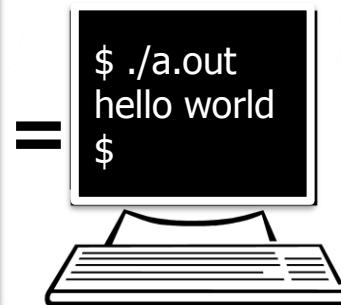
Object file
(hello.o)

```
01.ad5d0 003c 0000 0000 0000 0000 0000 0000  
01.ad5e0 0000 0000 03f2 0000 0001 0000 0000  
01.ad5f0 0000 0000 c6c0 001a 003b 0000 0000  
01.ad600 0000 0000 0020 0000 0000 0000 0000  
01.ad610 0001 0000 0000 0000 0000 0000 0000  
01.ad620 003b 0000 0000 0000 0000 0000 0000  
01.ad630 0000 0000 0430 0000 0001 0000 0000  
01.ad640 0000 0000 c740 001a 0030 0000 0000  
01.ad650 0000 0000 0020 0000 0000 0000 0000  
01.ad660 0001 0000 0000 0000 0000 0000 0000  
01.ad670 0030 0000 0000 0000 0000 0000 0000  
01.ad680 0000 0000 045a 0000 0001 0000 0000  
01.ad690 0000 0000 c7c0 001a 0031 0000 0000  
01.ad6a0 0000 0000 0020 0000 0000 0000 0000  
01.ad6b0 0001 0000 0000 0000 0000 0000 0000  
01.ad6c0 0014 0000 0000 0000 0000 0000 0000  
01.ad6d0 0000 0000 0001 0000 0003 0000 0000  
01.ad6e0 0000 0000 c805 001a 047f 0000 0000  
01.ad6f0 0000 0001 0000 0000 0000 0000 0000
```

Standard C library
(libc.so or libc.a)

Compile

Linking



Result

C System

- **C language, preprocessor, compiler, library, and tools**
- **Header files**
 - Specifies C standard library functions
 - rand(), printf(), scanf(), strcpy(), ...
 - Usually contain only function prototype, NOT function code
 - Then, where are the codes?
 - Standard library (libc) is linked automatically
 - You should specify other libraries (math)

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int i, n;

    printf("\n%s\n%s",
        "Some randomly distributed integers will be printed.",
        "How many do you want to see? ");
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        if (i % 10 == 0)
            putchar('\n');
        printf("%7d", rand());
    }
    printf("\n\n");
    return 0;
}
```

gcc Compiler

- **GCC: GNU Compiler Collection**
- **gcc OR g++ are popular**
 - C++ is a superset of C
- **They call**
 1. preprocessor
 2. compiler
 3. assembler
 4. Linker

* The **GNU Project** is a free software, mass collaboration project, announced on September 27, 1983, by Richard Stallman at MIT - wikipedia

gcc Options

- perform parts of 4 steps
- specify output files and format; “`gcc xxx.c -o x`”
- `-ansi` OR specify differences
- warning options “`-w`”
- debugging options “`-g`” “`-ggdb`” “`-g3`”
- optimization “`-O0`” “`-O2`”
- preprocessor options
- assembler options
- linker options “`-l`”
-

Summary

■ C program

- Preprocessing
- Variables, expressions, and assignments
- Input/Output
- Control flow
- Functions
- Arrays, strings, and pointers
- Files

■ Lexical elements, operators and C system

- Lexical analysis
- Comments, keywords, identifiers, constants, string constants
- Operators and precedence and associativity rules of them