System-oriented Programming, Prof. Philippe Cudré-Mauroux, Michael Luggen

Handout

Series S02:
- KR's Chap. 2
- Memory
- scanf

1. Prologue

For all exercises where you have to test your solution on machine (i.e. exercises 10c and 13b), a template is available on Moodle.

2. Reading

a) Browse/Read [KR88], chap. 2 : "Types, Operators and Expressions".

b) Study the lecture notes, and the tutorials TC01 "C Tutorial", TC02 "scanf" and TS02 "Memory Model I".

PART 1. C

3. First C Contact

Do the TC01 "C Tutorial", available at on Moodle.

Remark: A very good knowledge of this tutorial is mandatory for the rest of this course! Play around a little and test one or two programs from [KR 88] chap. 1 .

4. Word Count - Some Shell Commands

The program kr020.c, available on Moodle inside [C01] , reads a file from the standard input, counts the number of lines, words and characters, and prints these numbers. Download it, save it with the name "wcount.c" and compile it :

```
% gcc -o wcount wcount.c
```

Test it and explain the following Unix commands (consult [TU01] or the man pages, if some Unix commands are not familiar to you):

```
% ./wcount                       // input from keyboard
% ./wcount < wcount.c            // input redirection to wcount.c
% ./wcount < wcount > test
% cat wcount.c | ./wcount        // pipe
% grep { wcount.c
% grep { wcount.c | ./wcount
% grep -l { * | ./wcount
```

EOF Reminder: In most Unix shells, the End-of-file character (EOF) is produced by hitting <return>, followed by <Ctrl-D>. If this doesn't work, try <return>, followed by <Ctrl-Option(alt)-D>.

5. Type Conversion, Casting and ASCII

What is the output of the C code excerpt of Fig. 1 ? Also explain, for each output, why an explicit casting is necessary or not. Hint: for line 1 and 2 consult the ASCII table.

```
char c='A'; int i=65; float pi=3.14;
printf("%c %i\n", c, c);
printf("%c %i\n", i, i);
printf("%f %i\n", pi, (int)pi);
```

Fig.1

6. Constant, Variable, Escape Character '\', and Octal resp. Hexadecimal Digits

What is the output of the C code excerpt of Fig. 2, and explain. Hint: consult the ASCII table.

```
#define AT '\100'  // constant in octal
char at = '\x40';  // variable in hexadecimal
printf("%c %i %o %x\n", '@', '@', '@', '@');
printf("%c %i %o %x\n", AT, AT, AT, AT);
printf("%c %i %o %x\n", at, at, at, at);
```

Fig. 2

7. enum Type

What is the output of the C code excerpt of Fig. 3, and explain.

```
enum {FALSE, TRUE} b;       // declaration of variable b, without tag
b = TRUE;
printf("%i %i\n", b, FALSE);

enum color_tag {RED, GREEN, BLUE}; // enum declaration,
                                   // with tag 'color_tag'
enum color_tag c1, c2, c3;         // declaration of variables ci
c1 = RED; c2 = c1+1; c3 = BLUE;
printf("%i %i %i\n", c1, c2, c3);
```

Fig. 3

8. Logical Expressions

Write down the truth tables of the C expressions of Fig. 4, and for line 2 also give the value of p at the end of the evaluation by a C compiler:

```
p || !q
p && (p == q)            // Beware: p == q
p && (p = q) || (p = !q) // Beware: p = q
```

Fig. 4

9. Conditional Expression

a) What is the output of the C code excerpt of Fig. 5, and explain.

```
int x=3, y=2;
printf("%i\n", (x < y) ? x : y);
```

Fig.5

b) This is a Ternary Operator [1], an operator taking three arguments. It is available in most programming languages. The use of such conditional expressions is somewhat disputed these days and should only be use for extremely simple cases. Look up the actual discussion on the internet and write down a positive as also a negative point regarding the use of this language construct.

10. Bitwise operator

a) Assume that x is an unsigned integer. Proof that “`x<<1`” and “`x>>1`”, are resp. equivalent to “`2*x`” and “`x/2`”. Hint: use the binary representation of `x`.

b) Is this result still true if "`x<<1`" generates an overflow (i.e. when the left most bit of x is 1) ? Hint: do first a proof for a 4-bit machine, and then generalize it for a n-bit machine, assuming that unsigned integer arithmetic operations are done in 2n modulo modus).

c) Write a function `unsigned long setbit(unsigned long x, int n);` that returns x with the nth bit set to 1. Test your answer on machine.

11. Oder of Evaluation

Explain why:

```
f() + g()
printf(“%i\n”, ++n, f(n));
```

can produce different results with different compilers.

PART 2. System

12. Memory

Study the Tutorial TS02 “Memory Model I”, especially Figures 4 to 8 (mentioned below as by Fig. S4 to S8). Then draw the memory content of the program excerpt in Fig. 8 at the end of line 4 (assuming a 32-bit architecture):

a) with symbolic names and values of the variables, cf. Fig. S8;

b) same as (a), but augmented with symbolic names and possible hexadecimal values of the addresses, cf. Fig. S6;

c) big and little endian, with symbolic memory cell and address values, cf. Fig. S5,

d) big and little endian, with hexadecimal memory cell and address values, cf. Fig. S4.

```
main() {
   int i=8;
   char c1='@', c2='A';
   char s[5]="@A";
}
```

Fig 8.

13. scanf

Study the Tutorial TC02 “scanf”.

a) **A simple program with input errors**. Look at the program excerpt in Fig. 9a. What are the values of str, i and d at the end of lines 1, 2 and 3 if the input is: `AA 33 55 ZZ 77 99` ? Explain why.

```
char str[3]; int i, d;
d = scanf("%s %i", str, &i);
d = scanf("%s %i", str, &i);
d = scanf("%s %i", str, &i);
```

Fig. 9a.

b) **Your own simple program**. Write a program which reads a character, an integer, a string and a float, and prints them to the standard output. *Test your answer on a machine.*

Hand in. Upload your answers on Moodle.

References

[KR88] B. Kernighan, D. Ritchie, The C Programming Language, 2nd Ed., Prentice Hall, 1988.

[C01] Moodle > Tutorial “C01_KR88”

[TC01] Moodle > Tutorial “TC01 C Tutorial”

[TC02] Moodle > Tutorial “TC02 scanf”

[TS01] Moodle > Tutorial “TS02 Memory Model I”

[1] https://en.wikipedia.org/wiki/%3F: