Sistemske programiranje - C preschool

Jernej Vičič

1.1.2018
Overview

1. The C preschool
   - The shortest program in C
   - The shortest program in C - variant
   - The shortest program in C - variant
   - Hello in C
   - Compile
   - Make

2. System calls
   - How do system calls work?
   - Important file functions
   - Blocking mode
What is C?

- C is the most used programming language in system programming.
- C syntax is similar/same as Java’s,
- Actually it was the other way around, Java’s syntax was taken after C/C++,
- My personal opinion: C is a nice language:)
The shortest program in C

```c
void main(void)
{
}
```
The shortest program in C - variant (ANSI C)

```c
int main(void){
    return(0);
}
```
main()
{
}

The shortest program in C - variant

main(){}
main ( void ){
    printf ("Hello!");
    return (0);
}
- Write/copy the example
- Compile the example with gcc
Make

- Control over big source projects and more...
- Examples:
  - Big project: Makefile.apertium
  - Titles: MakefileTitles
  - Small project: MakefilePrimer1
System calls

- Standard C Library
- Application Programming Interface (API) to System-Calls
How do system calls work?

Kako delujejo sistemske klice

C Runtime Library

Program

Operacijski sistem

Jedro

Gonilnik

Jernej Vičič

Sistemske programiranje - C preschool
Important file functions

- int open(char *pathname, int flags);
- int read(int fd, void *buf, size_t count);
- int write(int fd, void *buf, size_t count);
- int close(int fd);
man

- hardcore interface, still usable,
- Example:
  man 2 open
Function open

- #include <fcntl.h>
- int open( const char *pathname, int flags );
- Translates path name into file descriptor,
- file-descriptor is a non-negative integer,
- Used as a file-ID in functions:
- flags are symbolic constants:
  - O_RDONLY, O_WRONLY, O_RDWR
Function read

- `#include <unistd.h>`
- `int read( int fd, void *buf, size_t count );`
- Tries to read `count` bytes,
- The read data is saved into variable ‘`buf’’ (memory-buffer),
- Returns number of read bytes,
- Returns -1 if there was an error,
- End of file (EOF) was reached if the return value is 0.
Function write

- `#include <unistd.h>`
- `int write(int fd, void *buf, size_t count);`
- Tries to write up to count bytes,
- Bytes are taken from variable ‘buf’ (memory-buffer),
- Returns number of written bytes,
- Returns -1 if there was an error,
- No data was written if it returns 0.
Some extra attention should be used in these examples for reading files - devices,

Function `read()` does not return 0, unless there the (EOF) was reached,

Devices should have data ready very soon,

On multiprogramming OS waiting is a big no-no!