Prerequisites

kaneton people

February 23, 2011

Outline

Outline

Description

The kaneton project is a very complex project which combined very different techniques, concepts, languages etc.

We will here quickly see the prerequisites to be sure students will be able to success.

List

The different prerequisites for the kaneton project are:

- Advanced makefiles because the kaneton compiling system uses special gmake features.
- C-preprocessor because the kaneton kernel uses it in a very powerful and elegant way.
- Inline assembly because it is widely used in low-level programming.
- C language because without, needless to start the project.
- Assembly language because low-level programming needs it.

The first three prerequisites will be coverded by dedicated courses.

For the two languages C and assembly, we assume you already well know these languages.

List

The different prerequisites for the kaneton project are:

- Advanced makefiles because the kaneton compiling system uses special gmake features.
- C-preprocessor because the kaneton kernel uses it in a very powerful and elegant way.
- Inline assembly because it is widely used in low-level programming.
- C language because without, needless to start the project.
- Assembly language because low-level programming needs it.

The first three prerequisites will be coverded by dedicated courses.

For the two languages C and assembly, we assume you already well know these languages.

List

The different prerequisites for the kaneton project are:

- Advanced makefiles because the kaneton compiling system uses special gmake features.
- C-preprocessor because the kaneton kernel uses it in a very powerful and elegant way.
- Inline assembly because it is widely used in low-level programming.
- C language because without, needless to start the project.
- Assembly language because low-level programming needs it.

The first three prerequisites will be coverded by dedicated courses.

For the two languages C and assembly, we assume you already well know these languages.

List

The different prerequisites for the kaneton project are:

- Advanced makefiles because the kaneton compiling system uses special gmake features.
- C-preprocessor because the kaneton kernel uses it in a very powerful and elegant way.
- Inline assembly because it is widely used in low-level programming.
- C language because without, needless to start the project.
- Assembly language because low-level programming needs it.

The first three prerequisites will be coverded by dedicated courses.

For the two languages C and assembly, we assume you already well know these languages.

List

The different prerequisites for the kaneton project are:

- Advanced makefiles because the kaneton compiling system uses special gmake features.
- C-preprocessor because the kaneton kernel uses it in a very powerful and elegant way.
- Inline assembly because it is widely used in low-level programming.
- C language because without, needless to start the project.
- Assembly language because low-level programming needs it.

The first three prerequisites will be coverded by dedicated courses.

For the two languages C and assembly, we assume you already well know these languages.

Outline

Overview

To be sure the C language is not a problem for every student we will view in this section some fundamental uses:

- Shifts.
- Bit masks.
- And more generally arithmetic and logic operators.

Shift Operators

- <<: shift left
- >>: shift right

Shift Operators

- <<: shift left
- >>: shift right

Example

The best examples of shift uses are either to quickly compute power of two:

```
int         v = 42

v << = 2;    /* now v is equal to 168 */
```

or to quickly set a bit from its position without any calculation:

```
#define FLAG_USER           0x1
#define FLAG_SYSTEM         0x2
#define FLAG_DRIVER         0x4
#define FLAG_SERVICE        0x8
```

instead we can do:

```
#define FLAG_USER           (1 << 0)
#define FLAG_SYSTEM         (1 << 1)
#define FLAG_DRIVER         (1 << 2)
#define FLAG_SERVICE        (1 << 3)
```

The result is a clearer output: we directly know which bit is set.

Logic Operators

- |: or operator.
- &: and operator.
- ^ : xor operator.
- ~ : not operator.

Logic Operators

- |: or operator.
- &: and operator.
- ^ : xor operator.
- ~ : not operator.

Logic Operators

- |: or operator.
- &: and operator.
- ^ : xor operator.
- ~ : not operator.

Logic Operators

- |: or operator.
- &: and operator.
- ^ : xor operator.
- ~ : not operator.

Example

```
unsigned int        set(unsigned int        mask,
                        unsigned char       bit)
{
  return (mask | (1 << bit));
}

unsigned int      unset(unsigned int        mask,
                        unsigned char       bit)
{
  return (mask & ~(1 << bit));
}
```

Outline

Overview

The assembly language is absolutely fundamental in low-level programming.

The kaneton project is essentially developed using the C language but small parts must be written in assembly either to optimise the source code but this is obviously useless and unwanted; or to develop very special parts in relation with the processor.

These architecture source code parts are essential and very difficult.

For these reasons the language must not be a difficulty.

Language

The assembly language will not be presented here because it depends of the architecture.

Please refer to the dedicated course on the architectures.

Outline

Overview

A thing students should understand is that in low-level programming there is no execution context.

This is very important because the students will have to create for example their own stack, to parse the ELF binary etc.

To do so, every student has to perfectly understand how an address space is composed.

Layout

1. **code**: .text
2. **data**: .data .rodata .bss
3. **heap**
4. **stack**
5. **kernel**

Stack Frames

Let's discuss about the stack internals and the stack frames.

The questions are:

- What does the compiler/programmer do?
- What does the processor do?