Chapter - <added> Portability Problems

Modularity

Stick as much machine dependent code into a single module.
When you change machines replace the module.

Word Size

The following works on 32-bit UNIX but fails on MS-DOS:

```
int zip;

zip = 92126;
std::cout << "Zip code " << zip << '\n';
```

It is non-portable.

On MS-DOS an **int** is 16 bits while on most UNIX systems it is 32.

Note: In the past you had to worry about 16 vs 32 bits. Today, *almost* everything is 32 bits. In the future you'll have to worry about 32 vs 64 bits.

Byte order problem

Motorola, Sun, HP order their bytes ABCD
Intel, DEC use BADC.

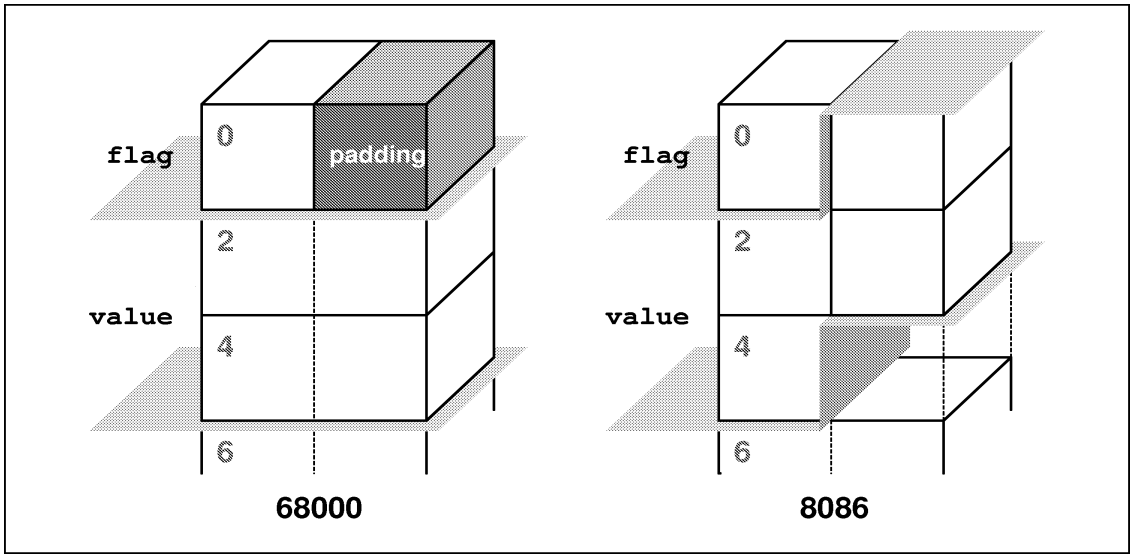
Writing 0x11223344 on a Sun and try to read it on an Intel machine you get 0x22114433.

One way around the byte order problem.

```
const int MAGIC      = 0x11223344; // file id number

// magic number byte swapped
const int SWAP_MAGIC = 0x22114433;

ifstream in_file;  // file containing binary data
long int magic;    // magic number from file

in_file.open("data");
in.file.read((char *)&magic, sizeof(magic));

switch (magic) {
    case MAGIC:
        // No problem
        break;
    case SWAP_MAGIC:
        cout <<"Converting file, please wait\n";
        convert_file(in_file);
        break;
    default:
        cerr << "Error:Bad magic number " << magic << '\n',;
        exit (8);
}
```

```
  following
     long int value; // value of the
  parameter
};
```

Alignment Problem




```
struct funny {
        char    flag;   // type of
data following
```

NULL Pointer problem

```
#define NULL 0

char *string;

string = NULL;
cout << "String is '" << str "'\n";
```

Note: This is actually illegal, but it’s frequently done.

File names

```
#ifndef __MSDOS__
#include <sys/stat.h> /* UNIX version of the file */
#else __MSDOS__
#include <sys\stat.h> /* DOS version of the file */
#endif __MSDOS__
```

Why does this program fail on MS-DOS/Windows?

Program output:

```
oot
ew      able:   file not found.
```

Program:

```
std::ifstream in_file;

#ifndef __MSDOS__
#define NAME "/root/new/table"
#else __MSDOS__
#define NAME "\root\new\table"
#endif __MSDOS__

in_file.open(NAME);
if (in_file.bad()) {
    std::cout << NAME << ": file not found\n";
    exit(8);
}
```

File Types

Some older versions of UNIX do not have `O_BINARY` defined.

```
#ifndef __MSDOS__
file_descriptor = open("file", O_RDONLY);
#else __MSDOS__
file_descriptor = open("file", O_RDONLY|O_BINARY);
#endif __MSDOS__
```

Better:

```
#ifndef O_BINARY  /* Do we have an O_BINARY? */
#define O_BINARY 0  /* If not, define these */
#define O_TEXT 0    /* so they don't get */
                    /* in the way */
#endif O_BINARY

. . .
file_descriptor =
                open("file", O_RDONLY|O_BINARY);
```

Porting four letter words

English:

Write a program to translate four letter words into more polite equivalents.

【練習問題 6】 テキストファイルの中で使われている言葉遣いを，きれいにするプログラムを書きなさい。このプログラムは，ファイル中に four-letter words(四文字語，卑猥な言葉, f***, c*** c*** s***d***, h*** など) を見つけたら，それをもっとおだやかな言葉に置き換えるものです。

(*** added)