Sistemes Operatius 2 - Pràctica 3

Taras Yarema

Arquitectures *big-endian* i *little-endian*

El terme de *endianness* [1] es refereix a l'ordre que segueixen els bytes en una arquitectura concreta, respecte a la representació binària de números. Els mètodes més comuns són el ordre *big-endian* i *little-endian*. La diferència entre aquests és que el primer, *big-endian*, ordena un número començant pels bytes **menys** significatiu, i el segon, *little-endian*, pels **més** significatius.

Per a posar un exemple concret, suposem que tenim el nombre hexadecimal `0x123456`. Necessitarem tres bytes per a poder representar-lo. En el cas d'un ordre *big-endian*, la representació, en octets, seria `0x12`, `0x34`, `0x56`. En canvi, amb la *little-endian*, seria `0x56`, `0x34`, `0x12`.

```
int n = 1337;
char *ptr = (char *)&n;

for (int i = 0; i < sizeof(n); i++)
    fwrite(ptr[i], 1, sizeof(char), f);
```

Listing 1: Escriure *big-endian*

```
int n = 1337;
char *ptr = (char *)&n;

for (int i = sizeof(n)-1; i >= 0; i--)
    fwrite(ptr[i], 1, sizeof(char), f);
```

Listing 2: Escriure *little-endian*

En conseqüència, això implica que a l'hora de llegir informació binària, s'ha de tenir en compte quin tipus d'ordre es segueix. En el cas de *big-endian*, s'hauran de llegir els bytes en l'ordre normal, de principi a fi. En canvi, en el cas de *little-endian*, l'ordre de lectura serà invers, de fi a principi.

Implicacions a nivell de memòria

El tipus de *endianness* afectaria a totes les dades que no siguin purament caràcters. És a dir, afectaria a les següents dades:

- *Magic number*: `int`;
- El nombre de nodes: `int`;
- Longituds de les paraules: `int`;
- El comptadors de cada paraula: `int`.

A les paraules no afectaria. Ja que estan formades per dades de tipus `char`, i es llegirien una a una en el ordre normal, des de la primera lletra fins a la última.

Compatibilitat

La solució que plantejo al problema de compatibilitat entre *endianness*, seria el següent. Utilitzar un un caràcter especial, byte, al començament del fitxer binari per a marcar el tipus de *endianness* del fitxer. Per exemple, es podria escriure un `0x00` si el format del fitxer és *big-endian*, i `0x01` si el format del fitxer és *little-endian*.

Concretament, el codi addicional seria, per a l'escritura, modificariem la funció `write_tree_file` al fitxer `write.c` i passaríem com a argument la *endianness* a la funció `write_node`, per a que sapiga com escriure els bytes.

```
char endianness = 0;
fwrite(&endianness, 1, sizeof(char), f);
...
if (tree->root != NIL)
    write_node(tree->root, f, endianness);
```

Listing 3: write.c

En el cas de la lectura, modificariem la funció `read_tree_file` al fitxer `write.c`.

```
char endianness;
fread(&endianness, 1, sizeof(char), f);

if (endianness == 0)
{
    // Read the file with
    // big-endian format
}
else if (endianness == 1)
{
    // Read the file with
    // little-endian format
}
else
{
    // We do not know about the endianness
    // exit with error
}
```

Listing 4: read.c

Referències

[1] Wikipedia: Endiannes.