CS 340
Computer Systems

#5: Endianness, Memory Hierarchy, and Virtual Memory

Sept. 6, 2022 · Wade Fagen-Ulmschneider

Endianness:
One major difference between ISAs is how multi-byte characters are stored. Knowing that **`sizeof(int) == 4`**, what do we expect from the following program?

05-endian.c

```
int i = 3 + (2 << 8) + (1 << 16);  // 66051
char *s = (char *)&i;
printf("%02x %02x %02x %02x\n", s[0], s[1], s[2], s[3]);
```

POWER - Big Endian:

s[0]	s[1]	s[2]	s[3]

ARM/x86-64 - Little Endian:

s[0]	s[1]	s[2]	s[3]

What is “Host Byte Order”? What is “Network Byte Order”?

Beyond Characters: Files and File Types
Using binary digits, often represented as characters using an encoding like UTF-8, we can build more complex file types.

File Extensions: An Easy Identifier
The most common way to identify the contents of a file is by the **file extension**. The file extension is defined as:

Examples:

cs340.png	**mp1.c**	**mp1.h**	**taylor.swift.mp4**

Which files are “plain text files”?

Memory Hierarchy:
The third foundation of a computer system is the “memory” -- the storage of data to be processed by our CPU. There are many different types of common **memory** and **storage** in a system:

1.

2.

3.

4.

5.

6.

Does knowing something about memory matter?

Sample Programs:

05-col.c

```
  for (unsigned int c = 0; c < SIZE; c++) {
    for (unsigned int r = 0; r < SIZE; r++) {
      array[(r * SIZE) + c] = (r * SIZE) + c;
    }
  }
```

-vs-

05-row.c

```
  for (unsigned int r = 0; r < SIZE; r++) {
    for (unsigned int c = 0; c < SIZE; c++) {
      array[(r * SIZE) + c] = (r * SIZE) + c;
    }
  }
```

...what is different about **`05-col.c`** and **`05-row.c`**?

Running Times: **`05-col.c`**:

`05-row.c`:

In working with memory in any computer system, we want to access it as quickly as possible. However, space is extremely limited in the fastest memory, so we need strategies on what data to keep close.

General Purpose Memory:

- CPU Registers:

- CPU Cache (i7-12700K, Released Q4'21):

- RAM:

Key Idea: Locality of Reference

System Memory:
Limited, Shared, and Simple

1.

2.

3.

To help us to begin to organize this RAM, we divide the RAM up into chunks called __________.

On Linux, find the size of a page:

```
# getconf PAGESIZE
```

...on almost every modern system, a page is ______ KB.

Virtual Memory:
Modern systems provide an abstraction between the _________ and ____________________:

1. A ____________________ translates a _________________ into a **physical address**.
2. Every memory address is made up of the _____________ and the _____________.
3. Virtual Memory is **NOT shared** between processes/apps.
4. **EVERY memory address** _______________________ is a virtual memory address!!

Virtual Memory Example:

Let's explore a sequence of allocations using a page table, where the physical RAM and every page table is 16 pages:

P1 Page Table:	RAM	P2 Page Table:	P3 Page Table:
[0]:	[0]:	[0]:	[0]:
[1]:	[1]:	[1]:	[1]:
[2]:	[2]:	[2]:	[2]:
[3]:	[3]:	[3]:	[3]:
[4]:	[4]:	[4]:	[4]:
[5]:	[5]:	[5]:	[5]:
[6]:	[6]:	[6]:	[6]:
[7]:	[7]:	[7]:	[7]:
[8]:	[8]:	[8]:	[8]:
[9]:	[9]:	[9]:	[9]:
[10]:	[10]:	[10]:	[10]:
[11]:	[11]:	[11]:	[11]:
[12]:	[12]:	[12]:	[12]:
[13]:	[13]:	[13]:	[13]:
[14]:	[14]:	[14]:	[14]:
[15]:	[15]:	[15]:	[15]:

Allocation Sequence:

1. Process #1 (P1): `a = malloc(3 * 4096)`
2. Process #3 (P3): `b = malloc(5 * 4096)`
3. Process #1 (P1): `c = malloc(2 * 4096)`
4. Process #3 (P3) exits.
5. Process #2 (P2): `d = malloc(4 * 4096)`
6. Process #2 (P2): `e = malloc(5 * 4096)`
7. Process #1 (P1): `a = realloc(a, 5 * 4096)`