CSIT 321 – Paradigm of Programming Languages A3-Tutorial

SUNY Fredonia

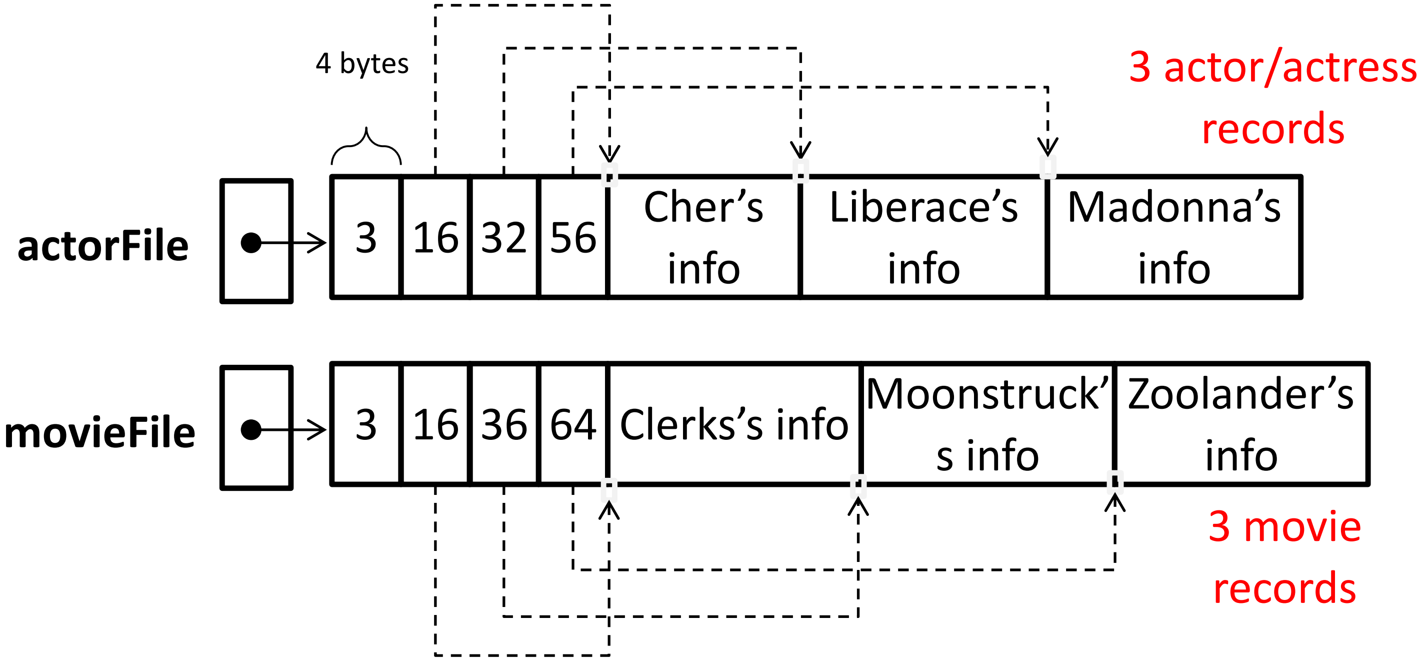
Gang Hu

```
class imdb {
  public:
    imdb(const string& directory);
    bool getPlayer(const size_t player_idx, vector<film>& films) const;
    bool getFilm(const size_t movie_idx, vector<string>& players) const;
    ~imdb();
  private:
    const void *actorFile;
    const void *movieFile;
};
```

Task

- getPlayer(size_t idx, vector<film>& films)

- getFilm(size_t idx, vector<string>& players)



4 bytes

actorFile	3	16	32	56	Cher’s info	Liberace’s info	Madonna’s info
movieFile	3	16	36	64	Clerks’s info	Moonstruck’s info	Zoolander’s info

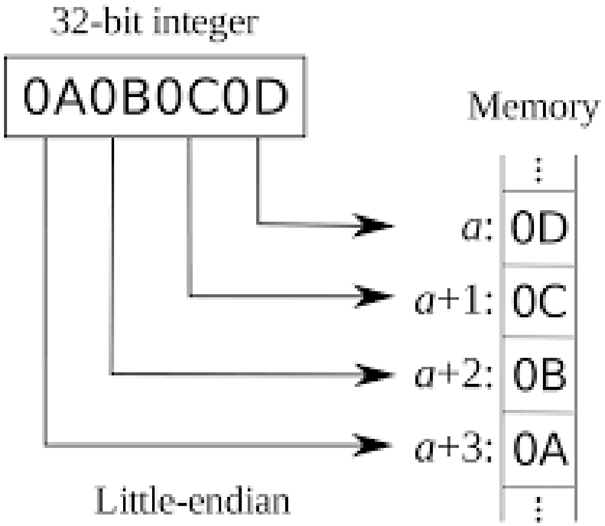
3 actor/actress records

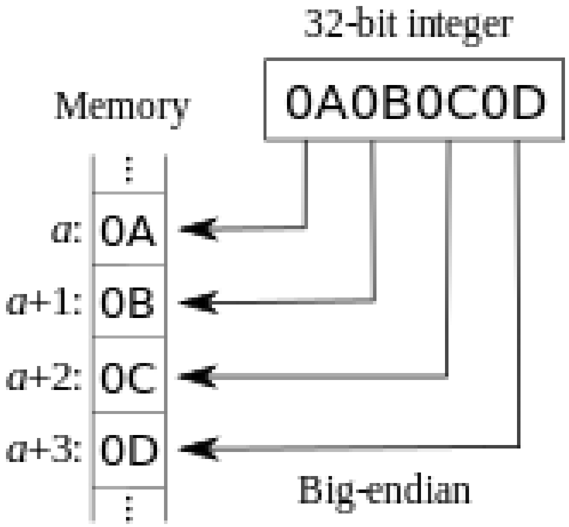
3 movie records

Big Endian vs. Little Endian

- Terms that describe the order in which a sequence of bytes are stored in computer memory
- Big Endian:
  – most-significant byte comes first.
    - Eg. decimal number 258
      – Binary format is 00000001 00000010

0x0000	***0x0001***	***Memory Address***
00000001	00000010	Big Endian
00000010	00000001	Little Endian


32-bit integer
0A0B0C0D
Memory
⋮
a: 0D
a+1: 0C
a+2: 0B
a+3: 0A
⋮
Little-endian


32-bit integer
0A0B0C0D
Memory
⋮
a: 0A
a+1: 0B
a+2: 0C
a+3: 0D
⋮
Big-endian

- Check first 4 bytes

```
 short test = 0;
 unsigned char c1 = (unsigned char)*(unsigned char*)actorFile;
//actorFile;
test = c1;
cout <<  test <<endl;
unsigned char c2 = (unsigned char)*((unsigned char*)actorFile + 1);
//actorFile;
test = c2;
cout << test << endl;
unsigned char c3 = (unsigned char)*((unsigned char*)actorFile + 2);
//actorFile;
test = c3;
cout << test << endl;
unsigned char c4 = (unsigned char)*((unsigned char*)actorFile + 3);
//actorFile;
test = c4;
cout << test << endl;
```

Bit Operation

- How to combine 2 unsigned chars into a16-bit value

B	1	1	1	1	0	0	0	0
A	0	0	0	0	1	1	1	1

  - H=A<<8;
  - V=H|B

H	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
V	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

- How to divide a 16-bit value into 2 unsigned chars
  - H=V>>8; // H=11110000
  - L=V&0x00FF; //L=00001111

V	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1

- Get total actors/actresses
  - we do not have to use bit operation to get data, since system has done it for us if it’s give right info

```
size_t n = (size_t)*(int*)actorFile; //actorFile;
```

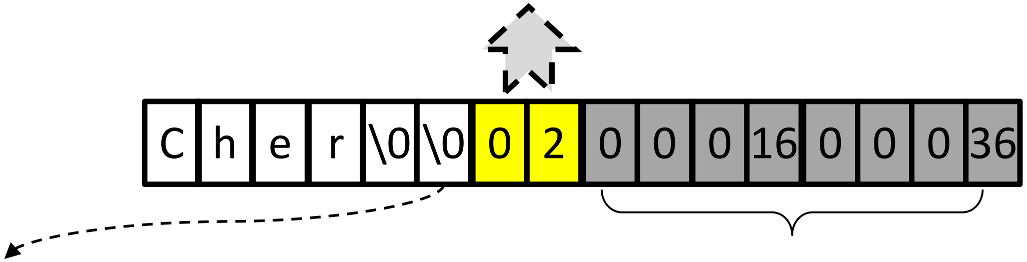
- Get start address of each actor/actress
  - *actorFile* is a pointer always pointing to the beginning of the memory space for the actor file

```
void *actor_adr = 0;
actor_adr = (void *)((int *)actorFile + 1 + idx);
```

Actor/Actress Record

***Cher*'s record: 16 bytes**

2 bytes storing the short int value 2, since Cher's starred in 2 movies. Since the total number of bytes occupied so far is a multiple of 4 (6+2), we don't need to pad with any additional \0



C	h	e	r	\0	\0	0	2	0	0	0	16	0	0	0	36

A padded byte, to make name size to be an even number

8 bytes are two 4-byte ints for offsets into the ***movieFile*** array.

Cher starred in 2 movies *Clerks* and *Moonstruck*, whose records are stored at 16 and 36 bytes from the base address of ***movieFile*** respectively.

- C string:
  - – Ended with null byte: '\0’
  - – strlen(“abc”)=3,
    - real space is 4 bytes

- Get a C string (actor/actress name string):

```
//base addr for a record
char * base = (char *)((char *)actorFile + *(int *)actor_adr);
int name_len = strlen((char *)base); //get C string length
```

- Get film number of this player

```
//current position with additional '/0'
 char *p = base + name_len + 1;
// check whether to pad or not
 if (name_len % 2 == 0){
   (char *)p += 1;
   name_len += 2;
  }
  else
   name_len++;

 unsigned short film_num = *(unsigned short *)p;
```

- Get film address offset

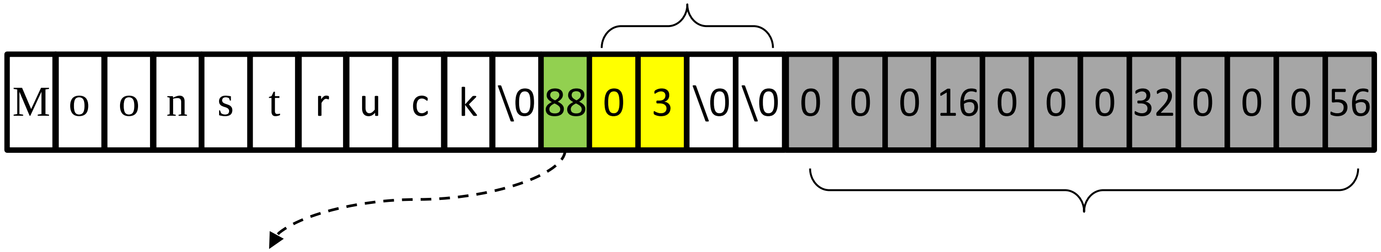
```
size_t current_size = p - base + 2;
//current_size includes name size and a 2-byte short for the film number, to see whether need to pad or not.
size_t zero_padding = current_size % 4;

p += zero_padding+2;
```

Movie’s Record

Moonstruck **’s record: 28 bytes**

- 2 bytes storing the short int value 3, since 3 actors starred in this movie.
- Since the total number of bytes occupied so far is 14, we need to pad with 2 additional \0 s, to make it as multiple of 4 (14+2)



M	o	o	n	s	t	r	u	c	k	\0	88	0	3	\0	\0	0	0	0	16	0	0	0	32	0	0	0	56

This movie was filmed in 1988. Since the name and year size is even (12), no padding is needed after this byte.

12 bytes are three 4-byte ints for offsets into the ***actorFile*** array.
3 actors in this movie: *Cher*, *Liberace, and Madonna*, whose records are stored at 16, 32 and 56 bytes from the base address of ***actorFile*** respectively.

- Get each movie’s base address
  - Add offset to *moviefile* pointer
- Get name
- Get year