1. The main memory is divided into some fixed sized blocks.
   These two will lead to internal fragmentation.

2. The main memory is divided into some variable sized blocks.
   These two will lead to external fragmentation and the operating system have to compact multiple holes and form a larger fragment.

3. A page is in the fixed sized block and a segment is in the variable sized block.
   Paging cause internal fragmentation and segmentation cause external fragmentation.
   Page address is generated by CPU and segmentation address is specified by the user.

4. Yes. They are similar but not exactly the same. The shared memory is located in the fixed address of the main memory. But in these two processes, the shared memory is bind into their own memory space i.e. virtual memory address. Therefore, the virtual memory addresses are not the same. However, the lower 12 bits of these two addresses should be the same because the page size is 4096 bytes. The start address of the shared buffer should be the start address of a page.
   Producer:

```
digongjiang@ubuntu:~/hw6$ ls
consumer.c  fib.c  fib.h  producer.c
digongjiang@ubuntu:~/hw6$ gcc fib.c producer.c -o producer -lrt
digongjiang@ubuntu:~/hw6$ ./producer 10
Start address of shared buffer: 7F1BB4800000
```

   Consumer:

```
digongjiang@ubuntu:~$ cd hw6
digongjiang@ubuntu:~/hw6$ ls
consumer.c  fib.c  fib.h  producer  producer.c
digongjiang@ubuntu:~/hw6$ gcc consumer.c -o consumer -lrt
digongjiang@ubuntu:~/hw6$ ./consumer 10
Start address of shared buffer: 7F9F4F4B9000
1
1
2
3
5
8
13
21
34
55
```

Within the relocatable object module, addresses of these two variables are both 0.

```
digongjiang@ubuntu:~/hw6$ gcc -c fib.c producer.c
digongjiang@ubuntu:~/hw6$ ls
consumer  consumer.c  fib.c  fib.h  fib.o  producer  producer.c  producer.o
digongjiang@ubuntu:~/hw6$ readelf -all fib.o | grep f0
000000000006  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
00000000001a  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
000000000028  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
00000000002e  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
00000000003c  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
00000000004a  000800000002 R_X86_64_PC32     0000000000000000 f0 - 4
     8: 0000000000000000     4 OBJECT  GLOBAL DEFAULT    4 f0
digongjiang@ubuntu:~/hw6$ readelf -all fib.o | grep f1
00000000000c  000900000002 R_X86_64_PC32     0000000000000000 f1 - 4
000000000014  000900000002 R_X86_64_PC32     0000000000000000 f1 - 4
000000000020  000900000002 R_X86_64_PC32     0000000000000000 f1 - 4
000000000034  000900000002 R_X86_64_PC32     0000000000000000 f1 - 4
000000000044  000900000002 R_X86_64_PC32     0000000000000000 f1 - 4
     9: 0000000000000000     4 OBJECT  GLOBAL DEFAULT    3 f1
```

In my code, these two variables are called f0 and f1. f0 is initialized by 0 and f1 is initialized by 1.

```c
#include "fib.h"
int f0=0,f1=1;
int fib()
{
    f1=f0^f1;
    f0=f0^f1;
    f1=(f0^f1)+f0;
    return f0;
}
```

When compiling, f0 is placed in .bss session, tagged 4 (global variable will be initialized by 0 after being defined), regarded as uninitialized variable. F1 is placed in .data session, tagged 3.

Within the absolute module the address of f0 and f1 are 0000000000202024 and 0000000000202010

```
digongjiang@ubuntu:~/hw6$ gcc fib.c producer.c -o producer -lrt
digongjiang@ubuntu:~/hw6$ readelf -all producer | grep f0
       00000000000000f0  0000000000000018  AI       5    22     8
  [13] .plt.got          PROGBITS         00000000000007f0  000007f0
 0x000000006ffffff0 (VERSYM)             0x4fc
000000201ff0  000c00000006 R_X86_64_GLOB_DAT 0000000000000000 _ITM_registerTMClo
neTa + 0
000000201fd0  000f00000007 R_X86_64_JUMP_SLO 0000000000000000 usleep@GLIBC_2.2.5
 + 0
    13: 00000000000007f0     0 SECTION LOCAL  DEFAULT   13
    45: 0000000000202024     4 OBJECT  GLOBAL DEFAULT   24 f0
digongjiang@ubuntu:~/hw6$ readelf -all producer | grep f1
    62: 0000000000202010     4 OBJECT  GLOBAL DEFAULT   23 f1
```