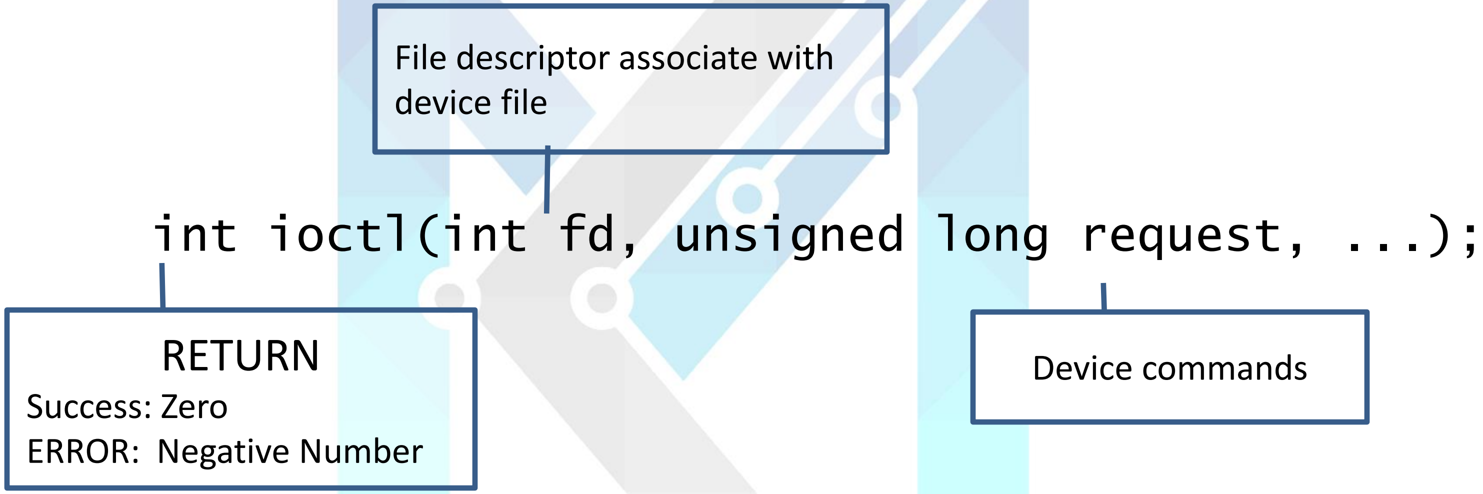
Advanced File operations

Advanced File Operations

Advanced file operations are overcome drawback of read() and write() system calls.

- **ioctl() :** read() & write() system calls doesn’t support to control device specific parameters. To control and get device specific parameters use ioctl().

ioctl() system call



File descriptor associate with device file

```
int ioctl(int fd, unsigned long request, ...);
```

RETURN

Success: Zero

ERROR: Negative Number

Device commands

Advanced File Operations

Advanced file operations are overcome drawback of read() and write() system calls.

- **select():** read() & write() system calls are blocking for single file descriptor select() system call blocks for multiple file descriptors rather than single file descriptor.

select() system call

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
            fd_set *exceptfds, struct timeval *timeout);
```

No. of File descriptor

readfds will be watched to see if characters become available for reading

writefds will be watched to see if space is available for write

RETURN

Success: No. of FD's contained in the three returned descriptor sets (that is, the total number of bits that are set in readfds, writefds, exceptfds)

TIMEOUT: Return ZERO

ERROR: Negative Number

exceptfds will be watched for exceptional conditions

should block waiting for a file descriptor to become ready

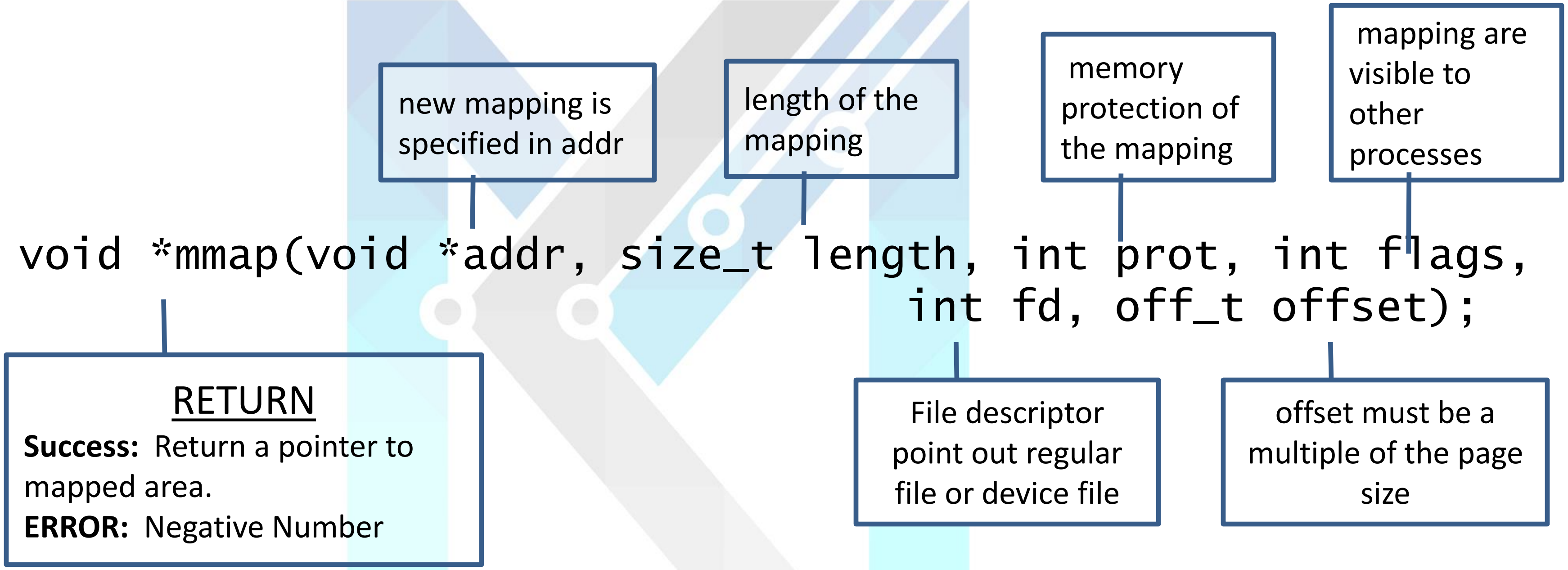
```
void FD_CLR(int fd, fd_set *set);
 int  FD_ISSET(int fd, fd_set *set);
 void FD_SET(int fd, fd_set *set);
 void FD_ZERO(fd_set *set);
```

Advanced File Operations

Advanced file operations are overcome drawback of read() and write() system calls.

- **mmap():** using mmap() system calls to map device buffers in to running process memory for better performance than read() and write() system calls

mmap() system call



new mapping is specified in addr

length of the mapping

memory protection of the mapping

mapping are visible to other processes

```
void *mmap(void *addr, size_t length, int prot, int flags,
                                  int fd, off_t offset);
```

RETURN

Success: Return a pointer to mapped area.

ERROR: Negative Number

File descriptor point out regular file or device file

offset must be a multiple of the page size

./mmap4 <file_name> <length> [offset]

```
./mmap4    test   5   4100
```

pa_offset = offset & ~(sysconf(_SC_PAGE_SIZE) - 1);
= 0x1004 & ~(0x1000-1)
= 0x1004 & ~(0x0FFF)
= 0x1004 & 0xF000
= 0x 1000 (page 1 base address)

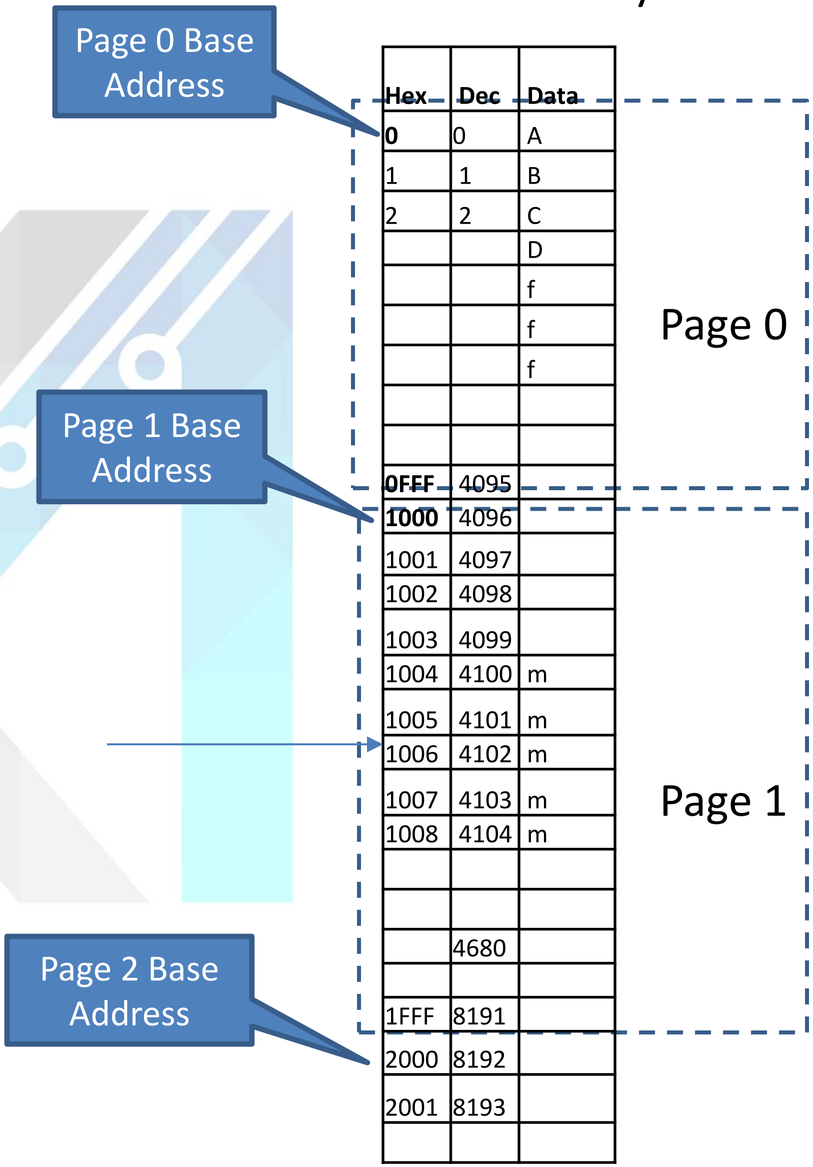
length = length + offset - pa_offset
= 5 + 4100 – 4096
= 9

= 0x0FFF & ~(4096-1)
= 0x0FFF & ~(0x0FFF)
= 0x0FFF & 0xF000 = 0x0000

mmap(NULL, 5, PROT_READ,MAP_PRIVATE, fd, pa_offset)

GUI -> CLI ; ctrl+alt+F1 ; ctrl+alt+F5 (2nd alt)
CLI -> GUI ; ctl+alt+F7 ; ctrl+alt+F2 (2nd alt)

test – 4681 Bytes



Hex	Dec	Data
0	0	A
1	1	B
2	2	C
		D
		f
		f
		f
0FFF	4095	
1000	4096	
1001	4097	
1002	4098	
1003	4099	
1004	4100	m
1005	4101	m
1006	4102	m
1007	4103	m
1008	4104	m
	4680	
1FFF	8191	
2000	8192	
2001	8193	