Lecture 10

Memory Management III

28 October 2011

Miscellaneous Left-overs

1. Optimal Page Size

s: average size of a process

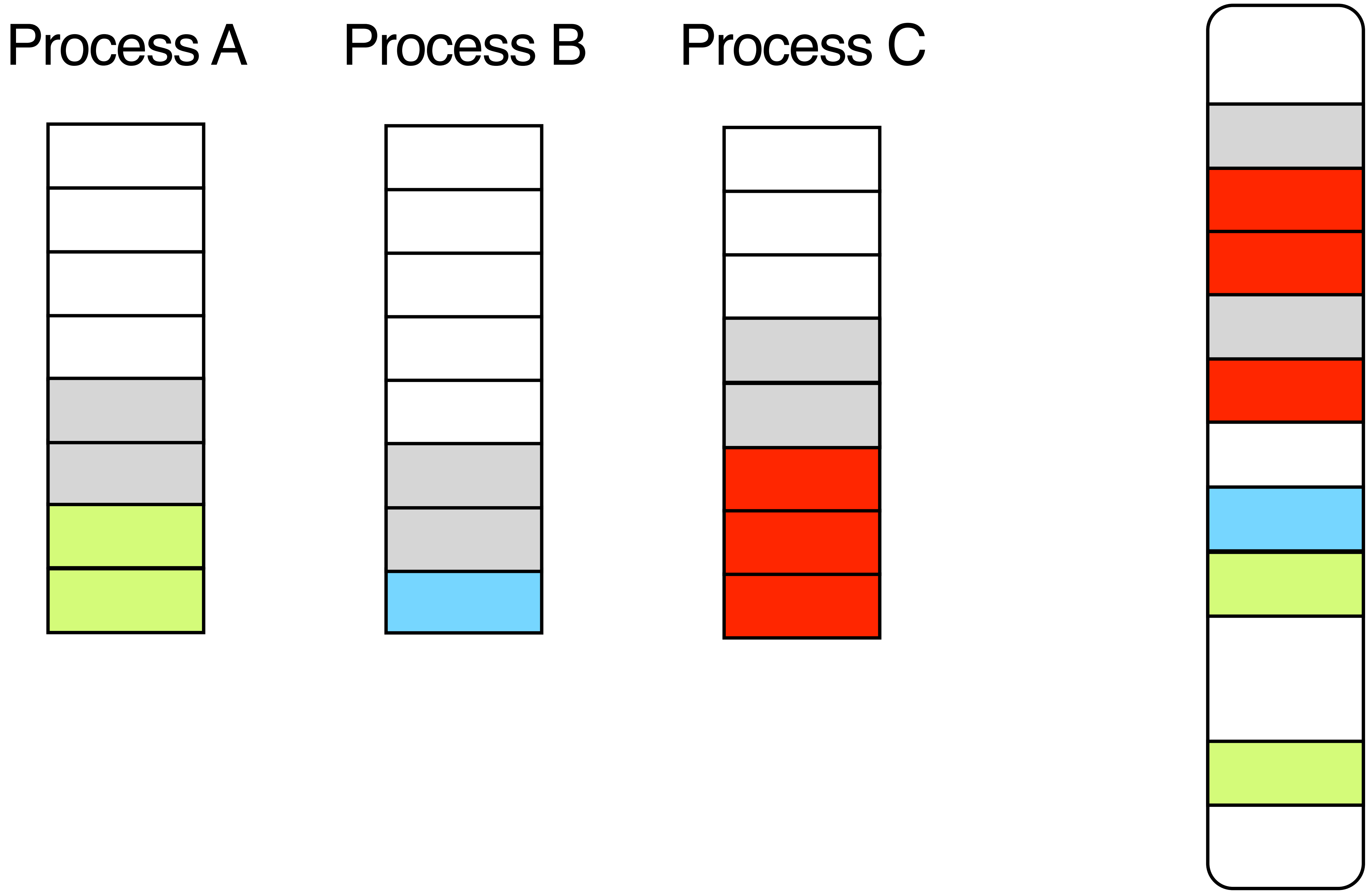
p: size of a page

e: size of a page entry

2. Shared Library

```
gcc savages.c -lpthread
gcc bush.c -lreadline
```

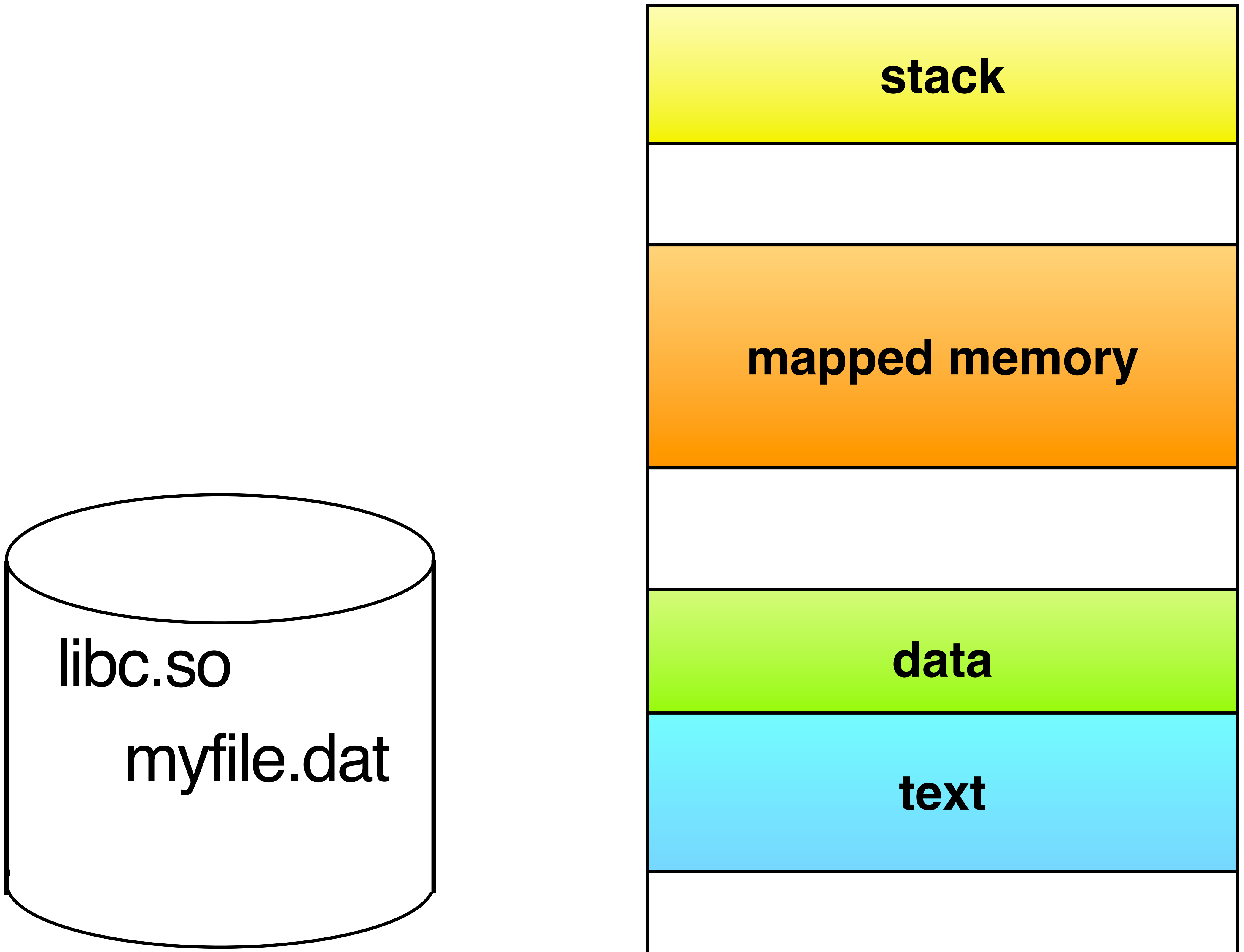
Physical Memory



position-independent code

3. Memory-mapped Files

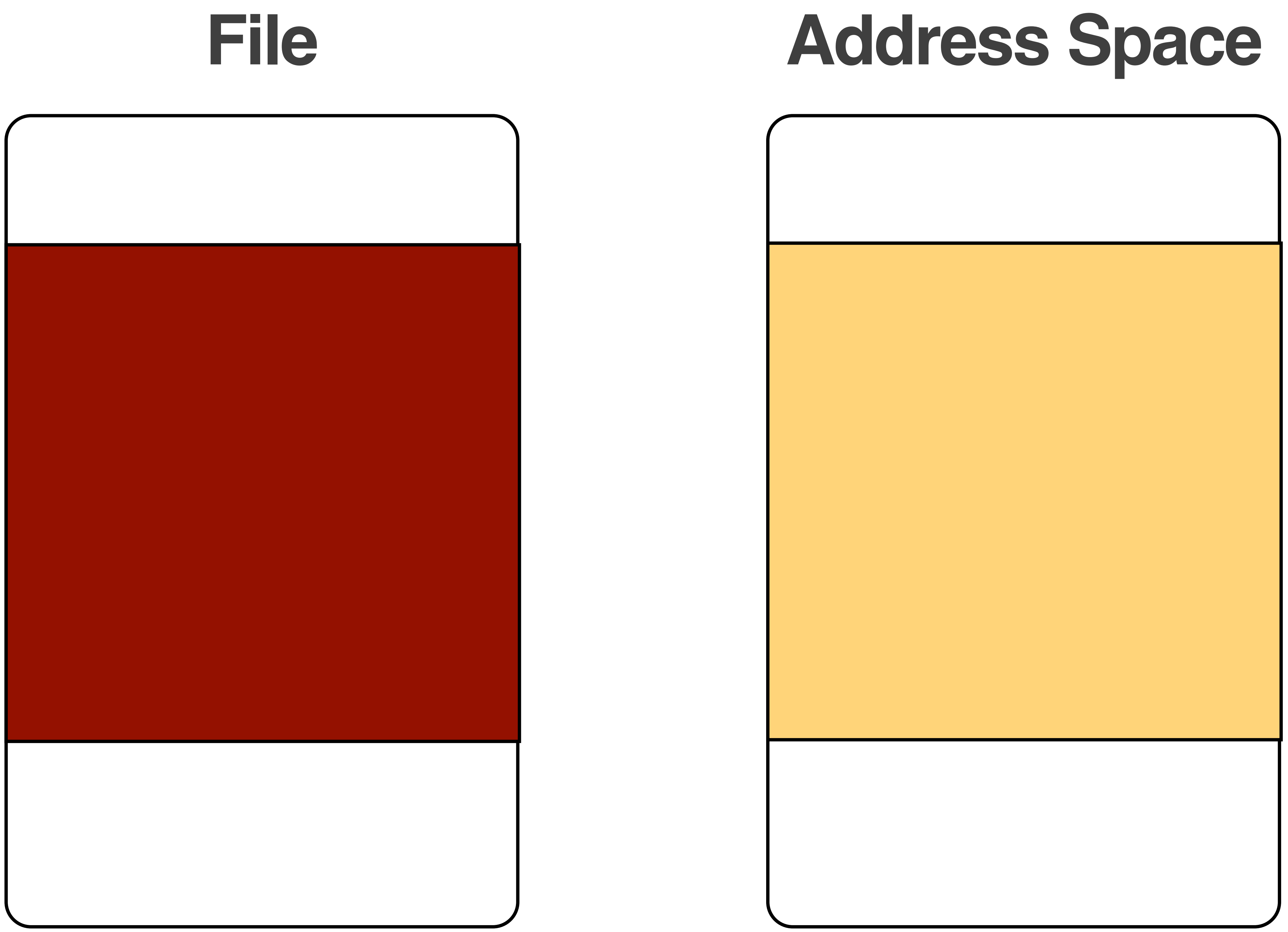
Address Space



load shared libraries
simplify file I/O
shared memory

System Calls

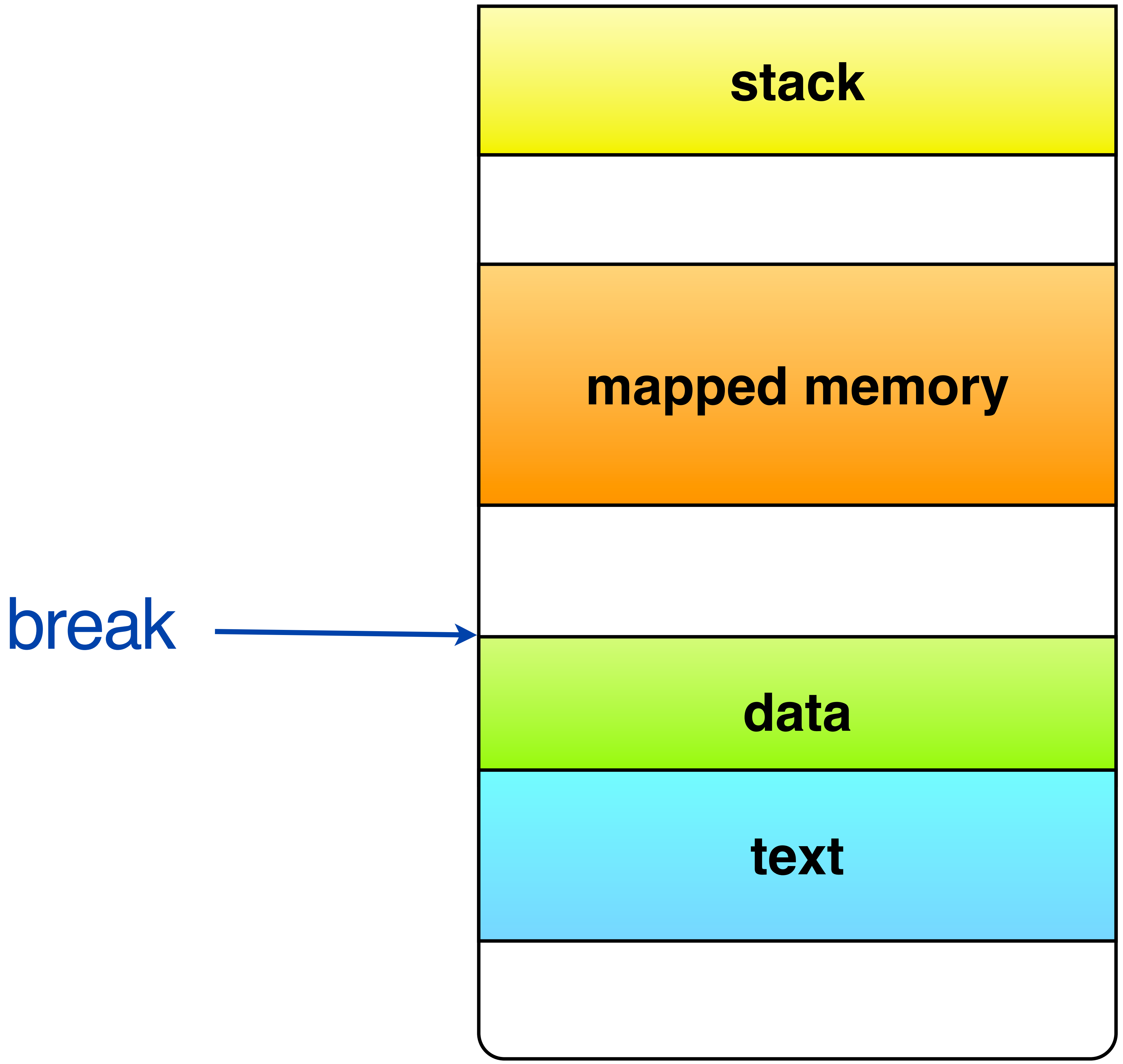
```
mmap()
munmap()
```



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

4. Dynamic Memory Allocation

Address Space



System Calls

`brk()`
`sbrk()`

(process request usable memory space from OS)

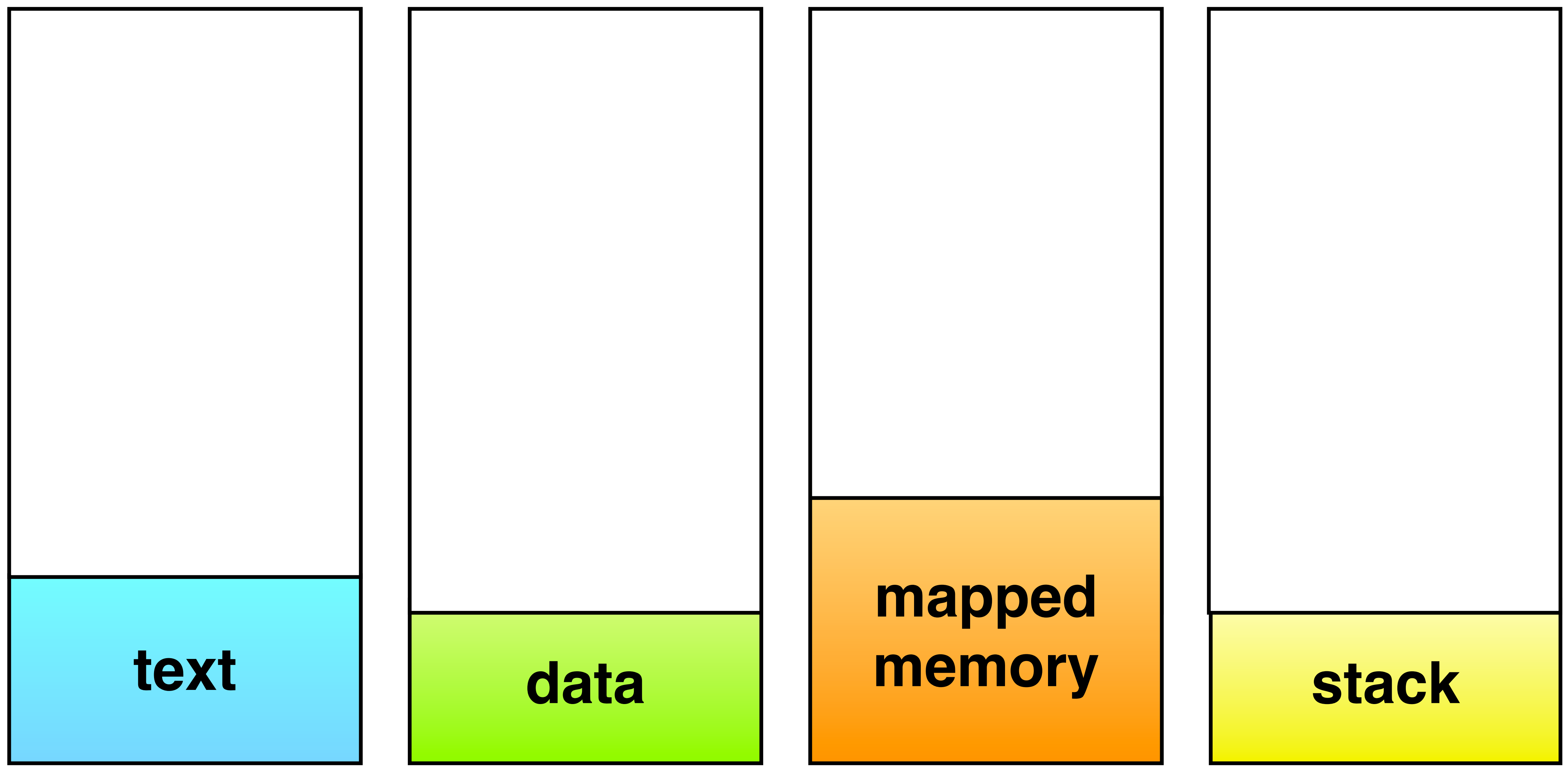
C library calls

```
malloc()
free()
```

Anonymous
Memory Maps

5. Memory Segmentation

Address Space per Segment



segment table

(segment id, base, limit, ..)

Advantages of Segmentation

1. separate protection
2. easier sharing

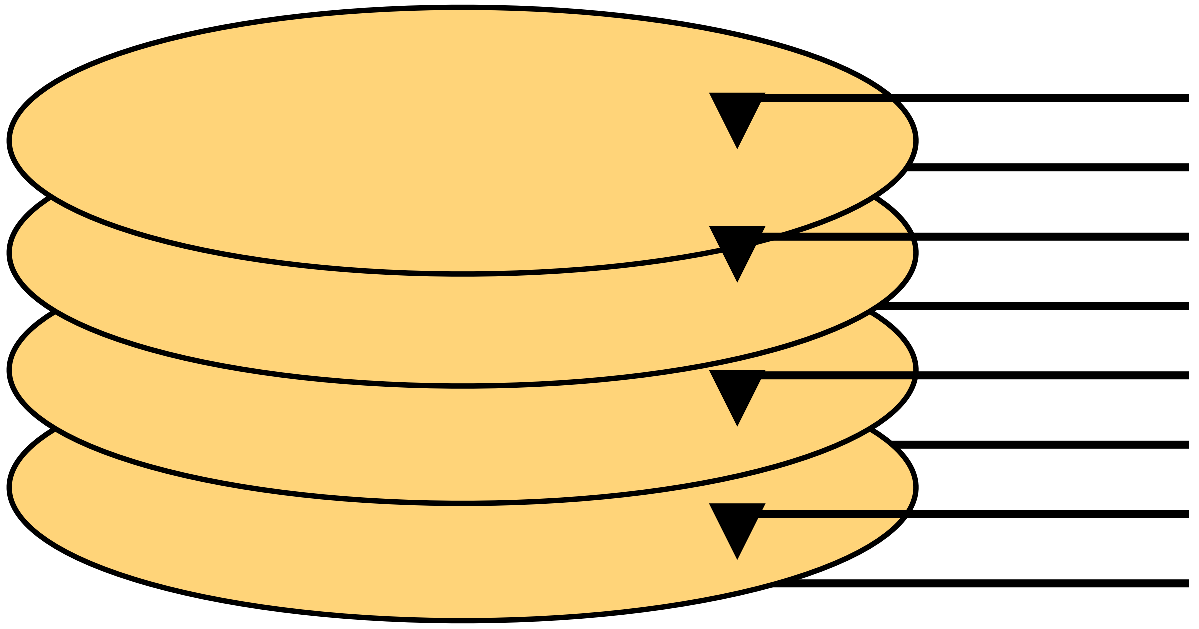
Lecture 10

File Systems

28 October 2011

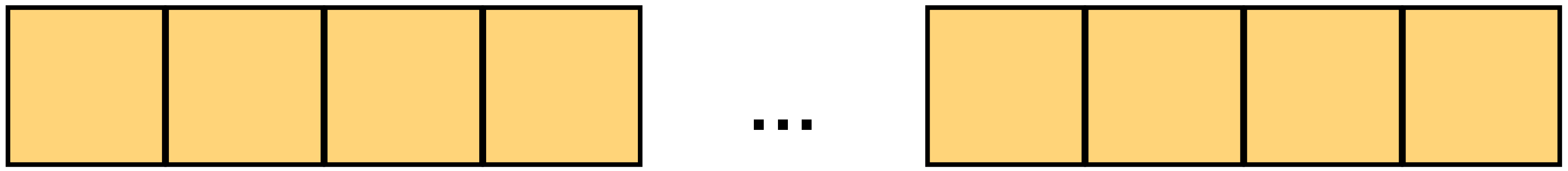
persistent storage
large storage
concurrent access

Magnetic Disks



seek, rotate, transfer

Disk Blocks



which data stored where?
which blocks are free?
who owns the block?

Abstraction: File System

directories

files

cwd

abs path

relative path

permissions

:

System Calls

creat	stat
open	fstat
close	pipe
read	dup
write	fcntl
lseek	

```
mkdir
rmdir
link
unlink
```

```
chdir
opendir
closedir
readdir
rewinddir
```

file descriptor table

special file descriptor

0

1

2

file system calls in action

(demo: strace cp foo bar)

a child process **inherit** the file descriptor table of its parent

system call: dup2()

system call: pipe()