

Vistula, IT Faculty, 2016

Operating Systems & Systems Programming

Dmitry A. Zaitsev

<http://daze.ho.ua>

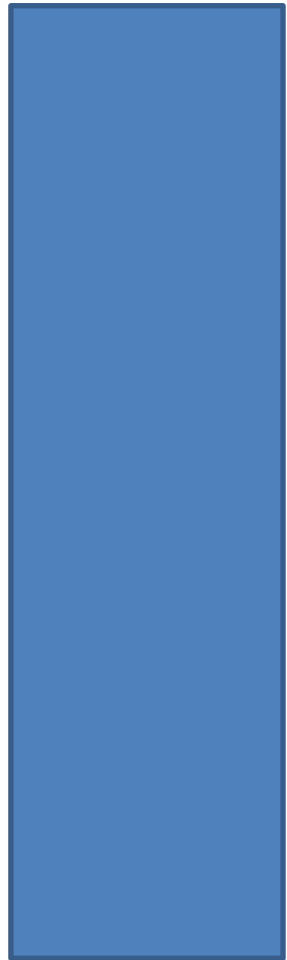
Lecture 5:

Virtual memory.

**Dynamic memory allocation and
shared segments.**

Virtual Memory Concept

Process virtual memory



Memory
mapping



Physical
RAM

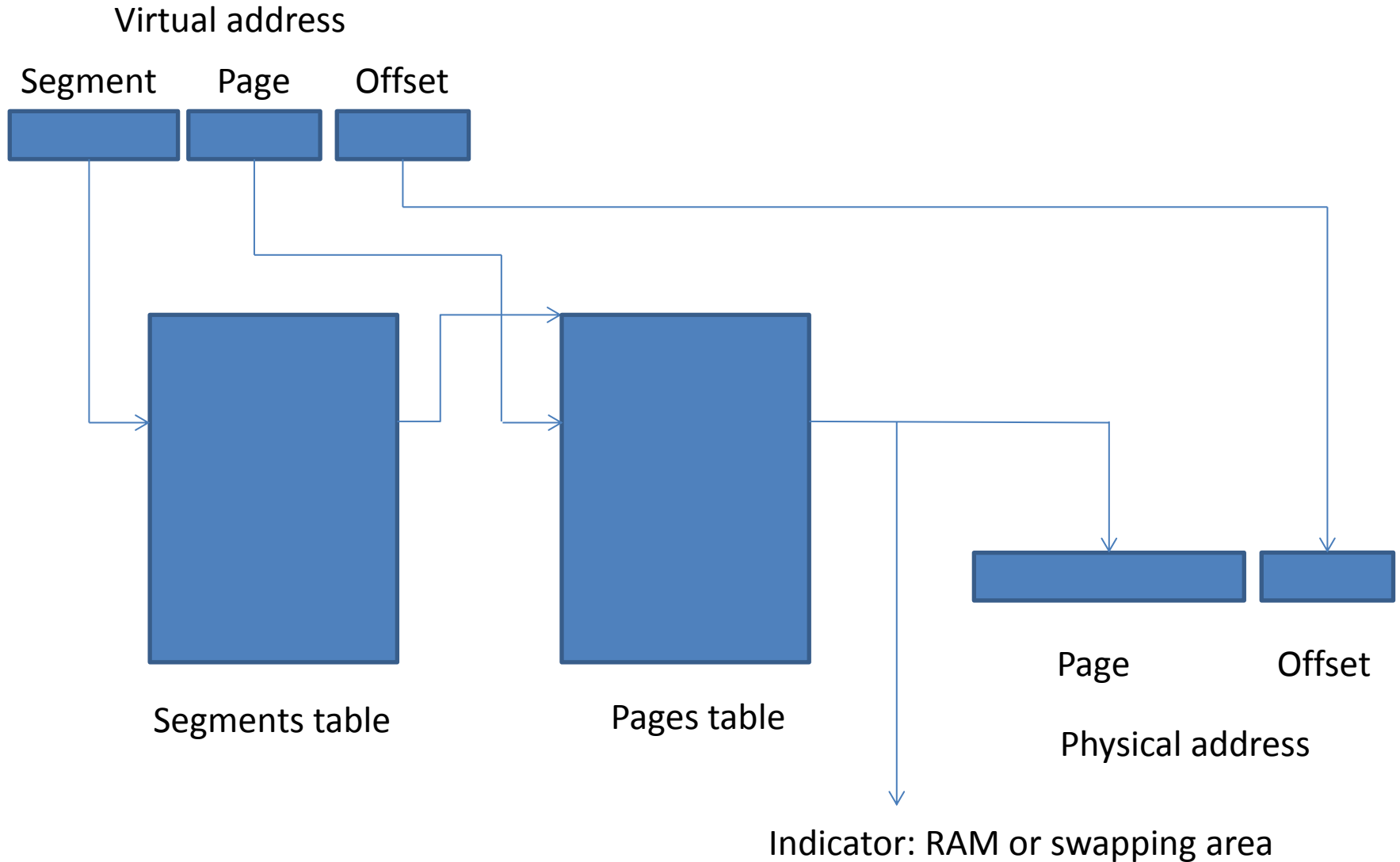


Swapping



Swapping
Area of HD

Segments and pages



Structure of executable file

- Header
- Segment(s) of code
- Segment(s) of data
- Stack
- Heap
- Shared segments

Types of memory in C program

- Static – declaration outside functions
- External – a reference
- Stack – declaration inside a function – created when entering the function and destroyed when exiting
- Dynamic – implicit allocation/deallocation with malloc/free – situate in heap

Allocate and free dynamic memory

- Allocate

```
void *malloc(size_t size);
```

- Free

```
void free(void *ptr);
```

- Allocate array

```
void *calloc(size_t nmemb, size_t size);
```

- Reallocate

```
void *realloc(void *ptr, size_t size);
```

Implementation of dynamic memory

Free sections
of heap



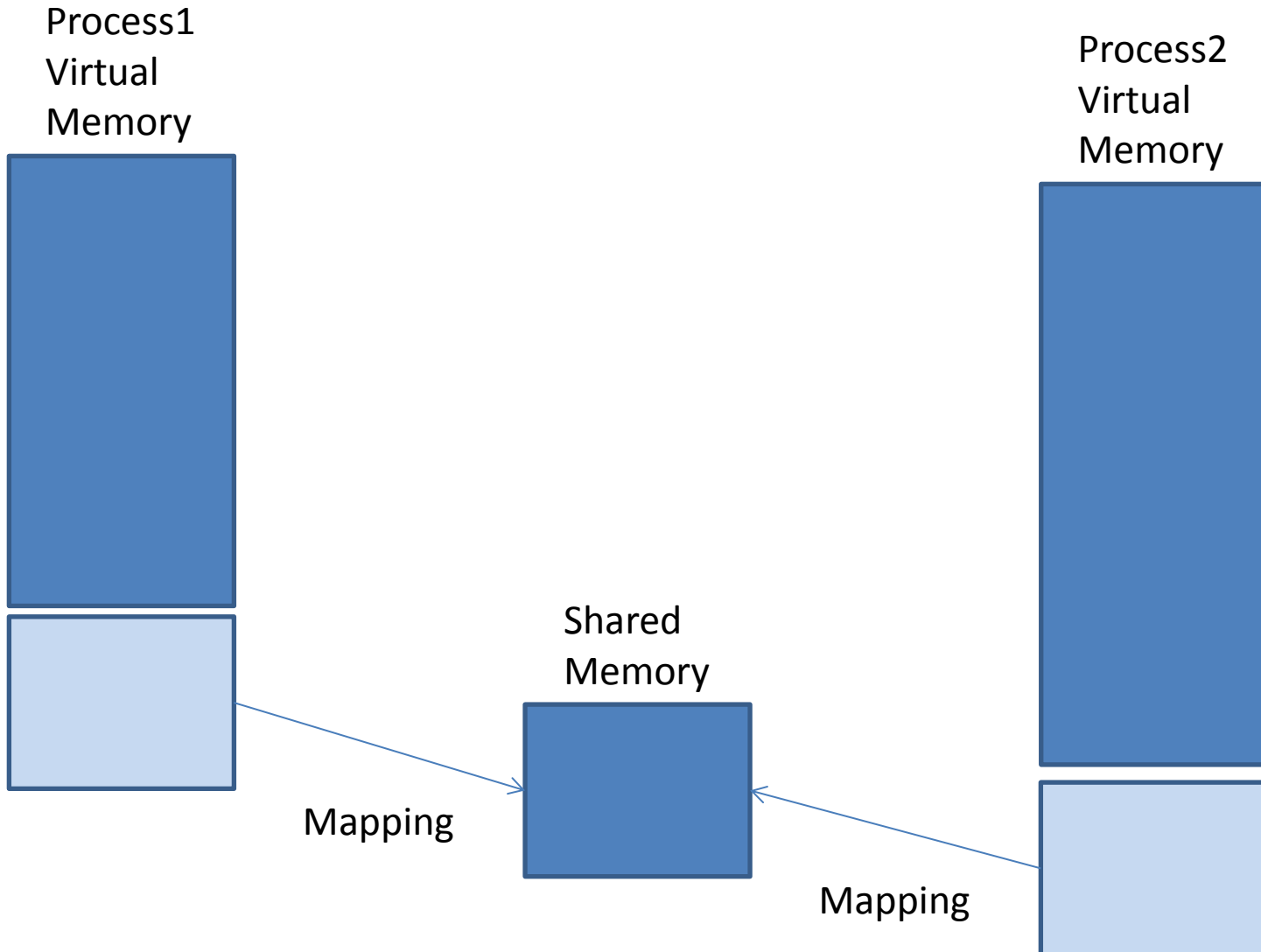
Allocated
sections
of heap



Section descriptor: address, length, attributes

Descriptor location – before a section

Shared memory concept



Shared memory segment

- `#include <sys/types.h>`

`#include <sys/shm.h>`

- Allocate a segment

`int shmget(key_t key, size_t size, int shmflg);`

- Attach a shared memory segment

`void *shmat(int shmid, const void *shmaddr, int shmflg);`

- Detach the shared memory segment

`int shmdt(const void *shmaddr);`

POSIX shared memory

- `#include <sys/mman.h>`
- Create/open shared memory object

```
int shm_open(const char *name, int oflag,  
mode_t mode);
```

- Unlink shared memory object

```
int shm_unlink(const char *name);
```

POSIX shared memory

- truncate a file to a specified length

```
int truncate(const char *path, off_t length);
```

```
int ftruncate(int fd, off_t length);
```

- map files or devices into memory

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

- unmap files or devices from memory

```
int munmap(void *addr, size_t length);
```

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27
main()
{
    char c;
    int shmid;
    key_t key;
    char *shm, *s;
    key = 5678;
    if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
        perror("shmget");
        exit(1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
    }
}
```

```
s = shm;

for (c = 'a'; c <= 'z'; c++)
    *s++ = c;
*s = NULL;

while (*shm != '*')
    sleep(1);

exit(0);
}
```

Server

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ 27
main()
{
    int shmid;
    key_t key;
    char *shm, *s;
    key = 5678;
    if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
        perror("shmget");
        exit(1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
    }
    for (s = shm; *s != NULL; s++) putchar(*s);
    putchar('\n');
    *shm = '*';
    exit(0);
}
```

Client

```

#include <semaphore.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>

int main(int argc, char **argv)
{
    int fd, i, count=0, nloop=10, zero=0, *ptr;
    sem_t mutex;
    fd = open("log.txt", O_RDWR | O_CREAT, S_IRWXU);
    write(fd, &zero, sizeof(int));
    ptr = mmap(NULL, sizeof(int),
        PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    close(fd);
    if( sem_init(&mutex, 1, 1) < 0)
    {
        perror("semaphore initialization");
        exit(0);
    }

```

```

    if (fork() == 0) { /* child process */
        for (i = 0; i < nloop; i++) {
            sem_wait(&mutex);
            printf("child: %d\n", (*ptr)++);
            sem_post(&mutex);
        }
        exit(0);
    }
    /* parent process */
    for (i = 0; i < nloop; i++) {
        sem_wait(&mutex);
        printf("parent: %d\n", (*ptr)++);
        sem_post(&mutex);
    }
    exit(0);
}

```

POSIX shared memory & semaphores