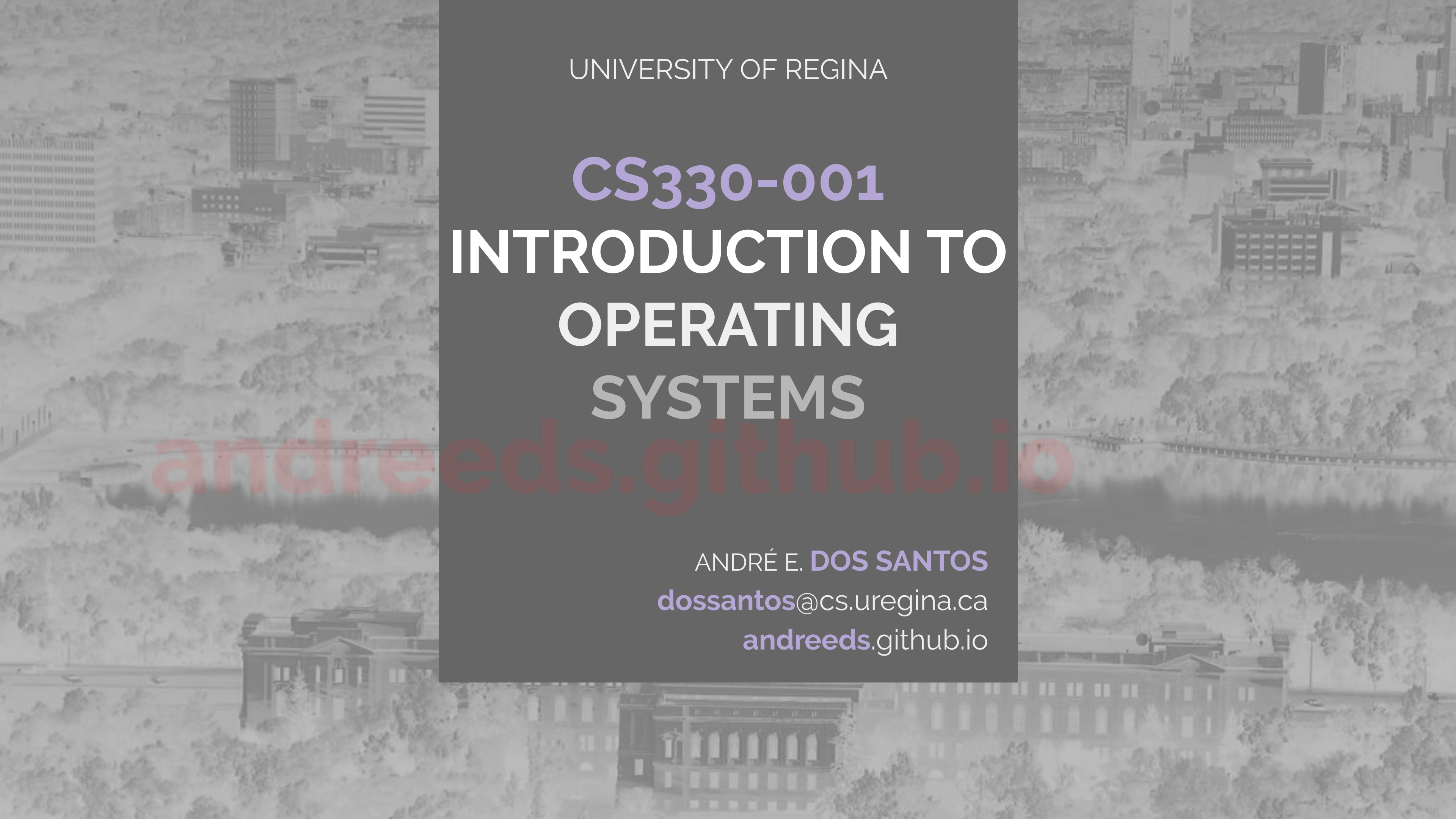


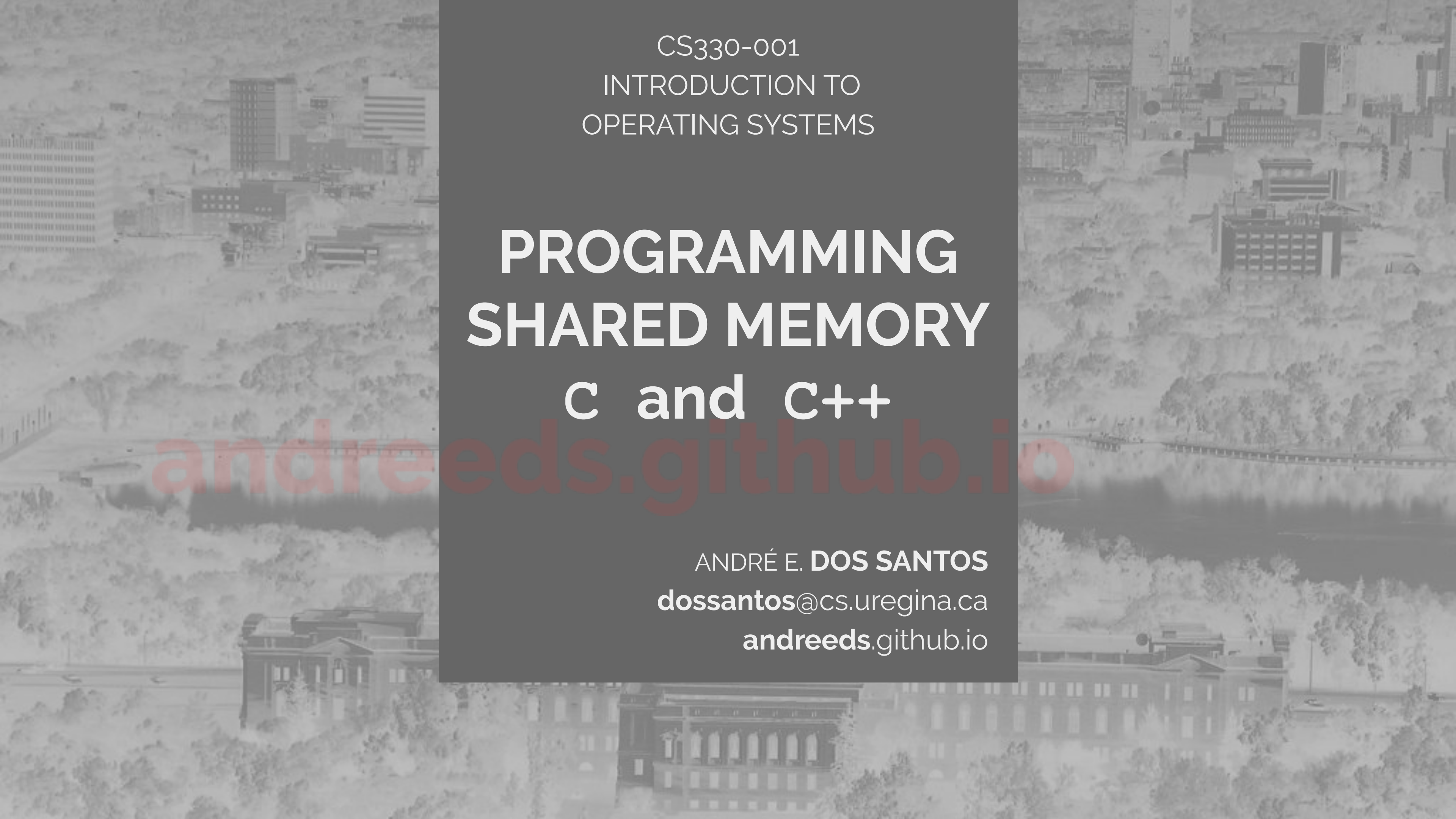
UNIVERSITY OF REGINA

CS330-001 INTRODUCTION TO OPERATING SYSTEMS

ANDRÉ E. DOS SANTOS

dossantos@cs.uregina.ca

andreeds.github.io



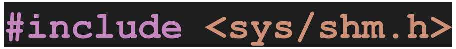
CS330-001
INTRODUCTION TO
OPERATING SYSTEMS

PROGRAMMING SHARED MEMORY `C` and `C++`

ANDRÉ E. **DOS SANTOS**
dossantos@cs.uregina.ca
andreeds.github.io

DATA STRUCTURES

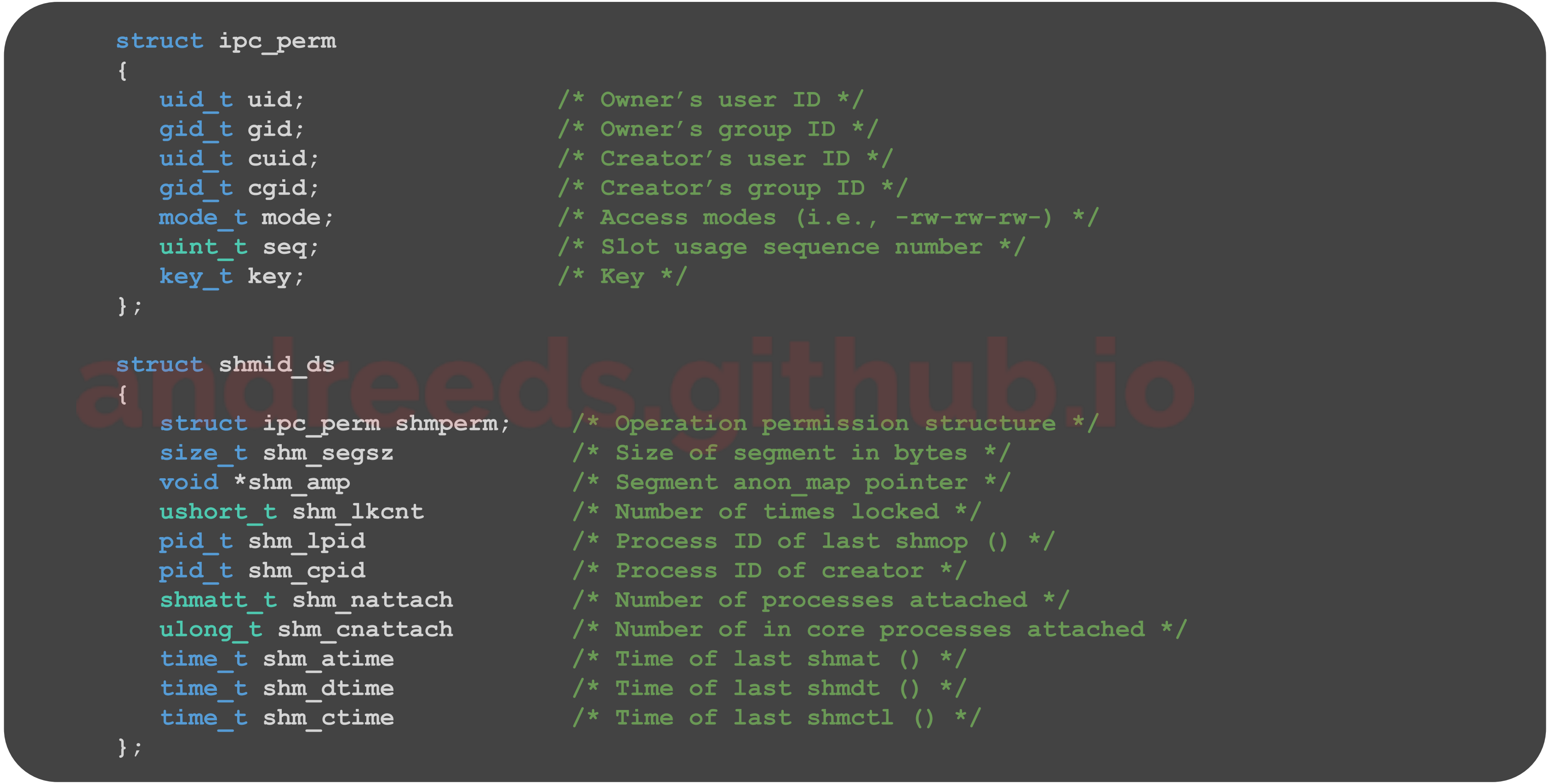
- A process maintains a data structure for every shared memory segment it is attached to



```
#include <sys/shm.h>
```

DATA STRUCTURES

- Part of this data structure keeps track of permissions (like those used for files)



```
struct ipc_perm
{
   uid_t uid;                    /* Owner's user ID */
   gid_t gid;                    /* Owner's group ID */
   uid_t cuid;                   /* Creator's user ID */
   gid_t cgid;                   /* Creator's group ID */
   mode_t mode;                  /* Access modes (i.e., -rw-rw-rw-) */
   uint_t seq;                   /* Slot usage sequence number */
   key_t key;                    /* Key */
};

struct shmid_ds
{
   struct ipc_perm shmperm;       /* Operation permission structure */
   size_t shm_segsz               /* Size of segment in bytes */
   void *shm_amp                  /* Segment anon_map pointer */
   ushort_t shm_lkcnt             /* Number of times locked */
   pid_t shm_lpid                 /* Process ID of last shmop () */
   pid_t shm_cpid                 /* Process ID of creator */
   shmatt_t shm_nattach           /* Number of processes attached */
   ulong_t shm_cnattach           /* Number of in core processes attached */
   time_t shm_atime               /* Time of last shmat () */
   time_t shm_dtime               /* Time of last shmdt () */
   time_t shm_ctime               /* Time of last shmctl () */
};
```

ELEMENTARY SYSTEM CALLS

The `shmget` system call is used to **create** a new, or **access** an existing, shared memory segment in kernel space

```
#include <sys/shm.h>
int shmget (key_t key, size_t size, int shmflags);
```

- The first parameter, `key`, designates the particular object to be created or accessed, and can be created by:
    - Letting the system pick the key (`IPC_PRIVATE`)
    - Picking the key “manually”.
- The second parameter, `size`, specifies the size of the memory segment required in bytes
- The third parameter, `shmflags`, specifies the access permissions for the shared memory segment
- If **successful**, `shmget` returns a non-negative **integer corresponding to the shared memory segment identifier** (kind of like a file descriptor) and initializes the shared memory segment to zero
    - If unsuccessful, shmget returns –1 and sets `errno`
- **A successful call to shmget allocates the shared memory, but it is not yet accessible**

ELEMENTARY SYSTEM CALLS

The `shmat` system call **attaches a shared memory segment to a process**

- i.e., maps the location of the shared memory segment in the kernel space into the memory space of a process

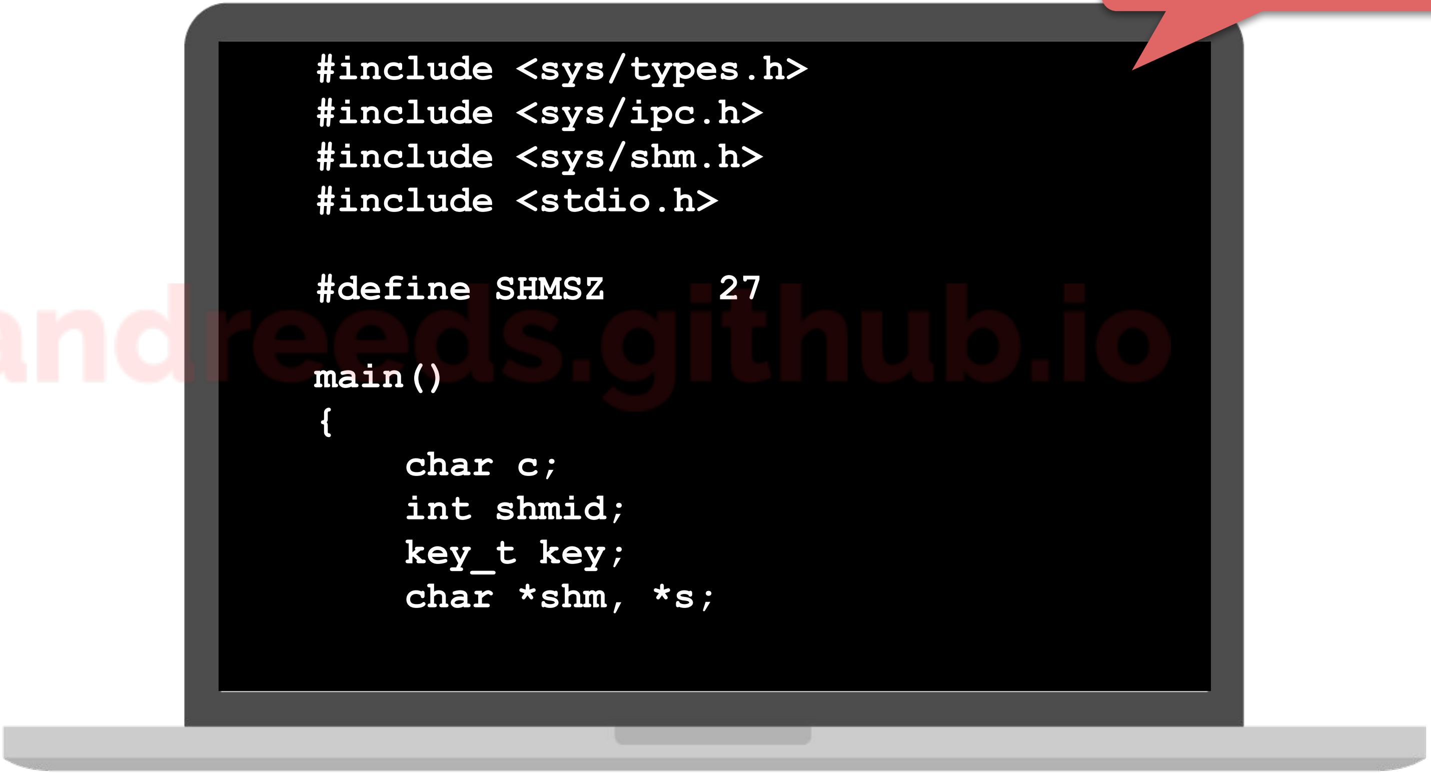
```
#include <sys/shm.h>
void *shmat (int shmid, const void *shmaddr, int shmflags);
```

- The first parameter, `shmid`, is the identifier for an existing shared memory segment
- The second parameter, `shmaddr`, determines the base address to which the shared memory segment will be attached
  - for our purposes, set this to **NULL** to let the kernel select the address
- The third parameter, `shmflags`, is used to define the behaviour of the shmat system call
  - for our purposes, set this to **0** to use the default permissions
- If **successful**, `shmat` returns the **starting address of the shared memory segment**.
  - If **unsuccessful**, `shmat` returns **–1** and sets `errno`
- **A shared memory segment can only be accessed after a successful call to shmat**

ELEMENTARY SYSTEM CALLS

shm_server.c AND shm_client.c

URCourses



```
#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;
```

SHELL COMMANDS

There are two shell commands for working with shared memory segments:

- `ipcs` : View the status of a shared memory segment
- `ipcrm` : Remove a shared memory segment

- Example – Removing a shared memory segment
  - ○ `ipcrm –m 151060502`
    - `151060502` is the `SHMID` from `ipcs` and is equal to segmentID

ELEMENTARY SYSTEM CALLS

The `shmdt` system call explicitly **detaches the shared memory segment from the address space of the calling process**

- however, when a process terminates, shared memory segments are implicitly detached

```
#include <sys/shm.h>
int shmdt (const void *shmaddr);
```

- If **successful**, `shmdt` returns **0**
  - If **unsuccessful**, shmdt returns **–1** and sets `errno`
- An `shmdt` system call does not remove the shared memory segment
- The last process to detach usually should remove the shared memory segment with an `shmctl` system call

ELEMENTARY SYSTEM CALLS

The `shmctl` system call performs a **control operation** on a shared memory segment

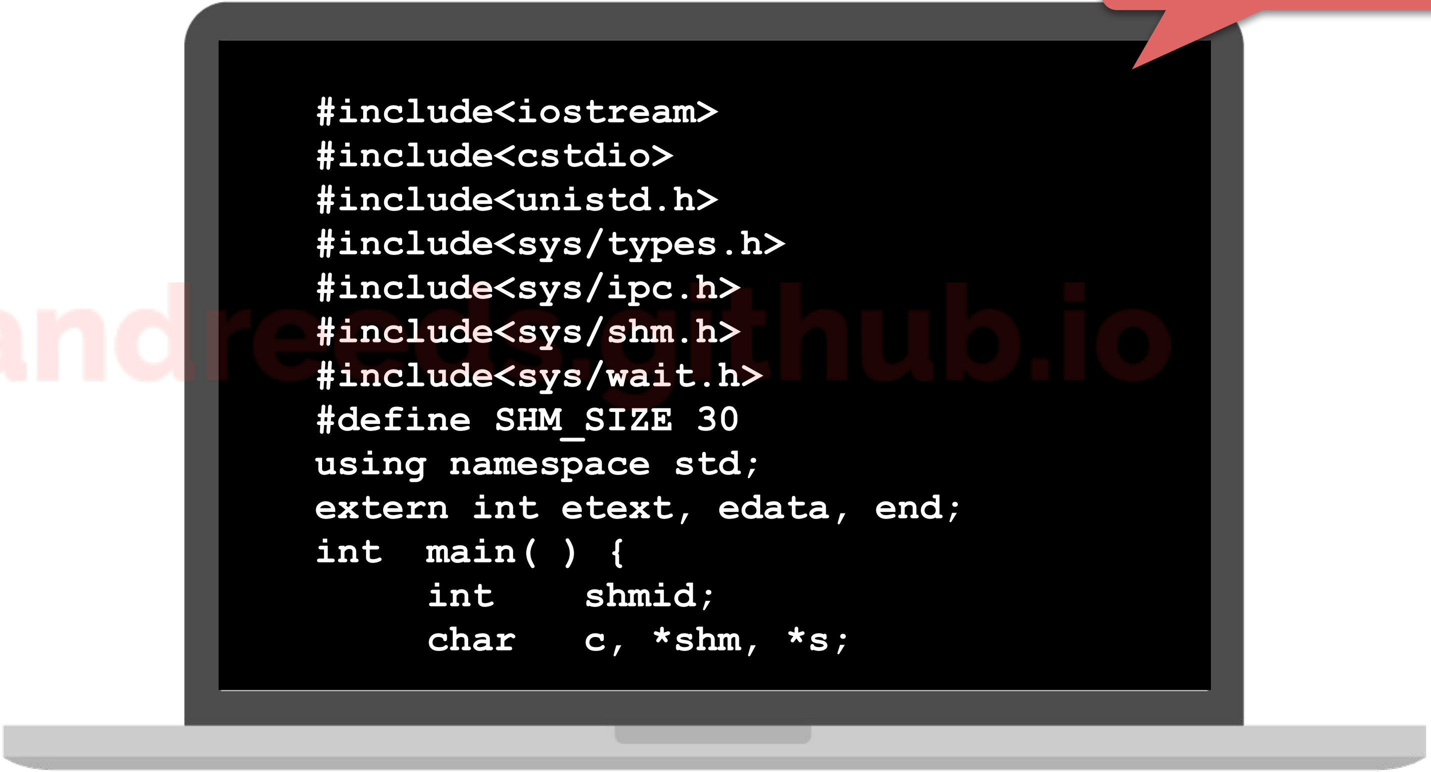
```
#include <sys/shm.h>
int shmctl (int shmid, int operation, struct shmid_ds *buffer);
```

- The first parameter, `shmid`, is the identifier for an attached shared memory segment
- The second parameter, `operation`, specifies the control operation to be performed
  - For our purposes, we are concerned only with:
    - `IPC_STAT`: Copies information from the kernel data structure associated with `shmid` into the `shmid_ds` structure pointed to by *buffer* (the caller must have read permission on the shared memory segment)
    - `IPC_RMID`: Marks the shared memory segment to be destroyed after the last process detaches from it (i.e., when the `shm_nattch` member of the associated structure `shmid_ds` is zero)
      - The caller must be the owner or creator of the segment, or be privileged
      - The buffer argument is ignored.
- There are other Linux-specific operations for copying and writing to buffer
- If **successful**, `shmctl` returns **0**
  - If **unsuccessful**, `shmctl` returns –1 and sets `errno`

ELEMENTARY SYSTEM CALLS

shmctl.cpp

URCourses



```
#include<iostream>
#include<cstdio>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/wait.h>
#define SHM_SIZE 30
using namespace std;
extern int etext, edata, end;
int  main( ) {
     int    shmid;
     char   c, *shm, *s;
```