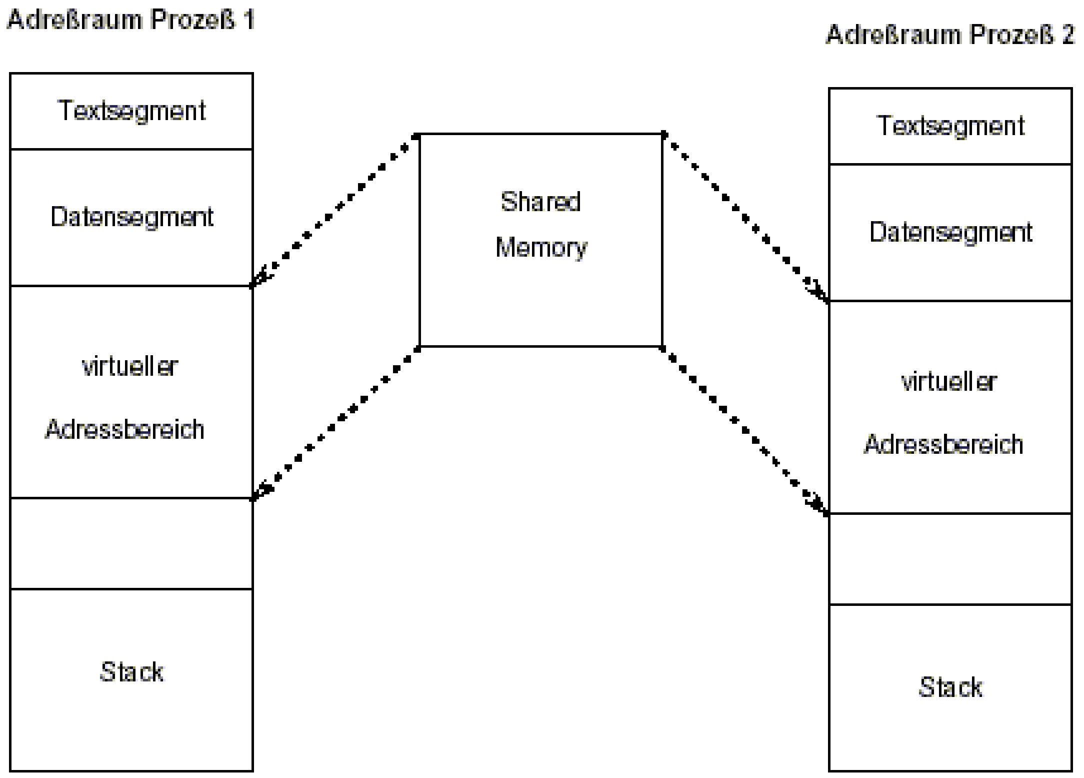
Using IPC: shared memory

Interprocess communication using shared memory

Lecturer: Erick Fredj

What is 'shared memory' ?

- Slice of memory accessible by arbitrary processes
- Pro: very fast
- Con: the access has to be synchronized, e.g. by using semaphores



How to use shared memory?

- How to use, step-by-step:

1. Allocate a shared memory segment: **shmget**()
2. Attach a shared memory segment to the address space of the calling process: **shmat**()
3. Writing into the shared memory, e.g. **memcpy**()
4. Detach the shared memory segment from the address space of the calling process: **shmdt**()
5. Set options, retrieve information on a shared memory segment: **shmctl**()

System call: shmget()

- shmget()
  - allocates a shared memory segment
  - returns the ID of the shared memory segment

Syntax:

```
int shmget( key_t key, int size, int shmflag );
```

Example:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
if(( shmid = shmget( 5L, 200, IPC_CREAT | 0666 )) < 0 )
{ perror("shmget failed"); exit(-1); }
```

System call: shmat()

- shmat()
  - attaches the shared memory segment to the address space of the calling process
  - Pointer supplied as return value, with which the common memory segment can be accessed

Syntax:

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

Example:

```
#include <sys/types.h>
#include <sys/shm.h>
if (( shm_p = shmat( shmid, (char *)0, 0 )) < (char *)0 )
{ perror("shmat failed"); exit(-1); }
```

Writing to the shared memory: memcpy()

- memcpy()
  - copies n bytes from source (src) to destination (dest)
  - returns pointer to the start of dest

Syntax:

```
void * memcpy( void *dest, void *src, size_t n );
```

Example:

```
#include <memory.h>
memcpy( shm_p, p_str, strlen(p_str) );
```

System call: shmdt()

- shmdt()
  - detaches a shared memory segment from the address space of the calling process
  - afterwards, no access to this memory segment possible (segmentation fault or core dump)

Syntax:

```
int shmdt( const void *shmaddr );
```

Example

```
#include <sys/shm.h>
if (shmdt( shm_p ) < 0 ) {
perror("shmdt failed"); exit(-1);
}
```

System call: shmctl()

- shmctl()
  - retrieves information about the shared memory segment (IPC_STAT)
  - sets permissions, owner, group (IPC_SET)
  - destroy a segment (IPC_RMID)

Syntax:

```
int shmctl( int shmid, int cmd, struct shmid_ds *buf );
```

Example:

```
#include <sys/ipc.h>
#include <sys/shm.h>
if( shmctl( shmid, IPC_RMID, 0) < 0 ) {
perror("shmctl failed"); exit(-1);
}
```

Shared memory and fork / exec / exit

- After a fork() the child inherits all the attached shared memory segments
- After an exec() all attached shared memory segments are detached (not destroyed)!!!
- When exit() all attached shared memory segments are detached (not destroyed)!!!