```
XSI IPC (based on SystemV IPC)

Three types of IPC: message queues, semaphores and shared memory

Handled using a non-negative integer identifier; Different processes can establish
communication using the same identifier.

A key (key_t) is used to establish a common identifier.
There are three possibilities to define a key:

1. IPC_PRIVATE: In this case an alternative channel to communicate the identifier
   between processes is necessary (parent/child fork, file system, ...).

2. A fixed predetermined key number (may collide with other existing keys).

3. ftok function to generate a key from a path and a byte integer.

Outside file system.  Handled in OS by ipcs and ipcrm (requires explicit removal, or reboot)

Usage:

- create new identifier: msgget/semget/shmget with key and IPC_CREAT and IPC_EXCL flags
- get existing identifier: msgget/semget/shmget with key and other arguments as zero


--------------------------------------------------------------------------------------------------------------------
--- Shared memory:
```

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

int shmid;

//creation:
shmid = shmget(key, size, 0600 | IPC_CREAT | IPC_EXCL);
if (shmid == -1)
{
    perror("Fail creating shared data");
    exit(EXIT_FAILURE);
}

//use existing:
shmid = shmget(key, 0, 0);
if (shmid == -1)
{
    perror("Fail creating shared data");
    exit(EXIT_FAILURE);
}

// attach shm to pointer address:
p = shmat(shmid, NULL, 0);
if (p == (void*)-1)
{
    perror("Fail connecting to shared data");
    exit(EXIT_FAILURE);
}

// detach shm from pointer address:
int st = shmdt(p);
if (st == -1)
{
    perror("Fail detaching shared data");
    exit(EXIT_FAILURE);
}

// destroy shm:
int st = shmctl(shmid, IPC_RMID, NULL);
if (st == -1)
{
    perror("Fail destroying shared data");
    exit(EXIT_FAILURE);
}
```

```
#include <process.h>

int shmid;

//creation:
shmid = pshmget(key, size, 0600 | IPC_CREAT | IPC_EXCL);

//use existing:
shmid = pshmget(key, 0, 0);

// attach shm to pointer address:
p = pshmat(shmid, NULL, 0);

// detach shm from pointer address:
pshmdt(p);

// destroy shm:
pshmctl(shmid, IPC_RMID, NULL);
```

```
------------------------------------------------------------------------------------------------------------------------
--- Semaphore:
```

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

//creation (1 semaphore):
semid = semget(key, 1, 0600 | IPC_CREAT | IPC_EXCL);
if (semid == -1)
{
    perror("Fail creating locker semaphore");
    exit(EXIT_FAILURE);
}

//use existing:
semid = semget(key, 0, 0);
if (semid == -1)
{
    perror("Fail creating shared data");
    exit(EXIT_FAILURE);
}

// destroy sem 0:
int st = semctl(semid, 0, IPC_RMID, NULL);
if (st == -1)
{
    perror("Fail destroying shared data");
    exit(EXIT_FAILURE);
}

// decrement:
struct sembuf down = {0, -1, 0};
if (semop(semid, &down, 1) == -1)
{
    perror("lock");
    exit(EXIT_FAILURE);
}

// increment:
struct sembuf up = {0, 1, 0};
if (semop(semid, &up, 1) == -1)
{
    perror("unlock");
    exit(EXIT_FAILURE);
}
```

```
#include <process.h>

//creation (1 semaphore):
semid = psemget(key, 1, 0600 | IPC_CREAT | IPC_EXCL);

//use existing:
semid = psemget(key, 0, 0);

// destroy sem 0:
psemctl(semid, 0, IPC_RMID, NULL);

// decrement:
struct sembuf down = {0, -1, 0};
psemop(semid, &down, 1);
// or simply:
psem_down(semid, 0);

// increment:
struct sembuf up = {0, 1, 0};
psemop(semid, &up, 1);
// or simply:
psem_up(semid, 0);
```