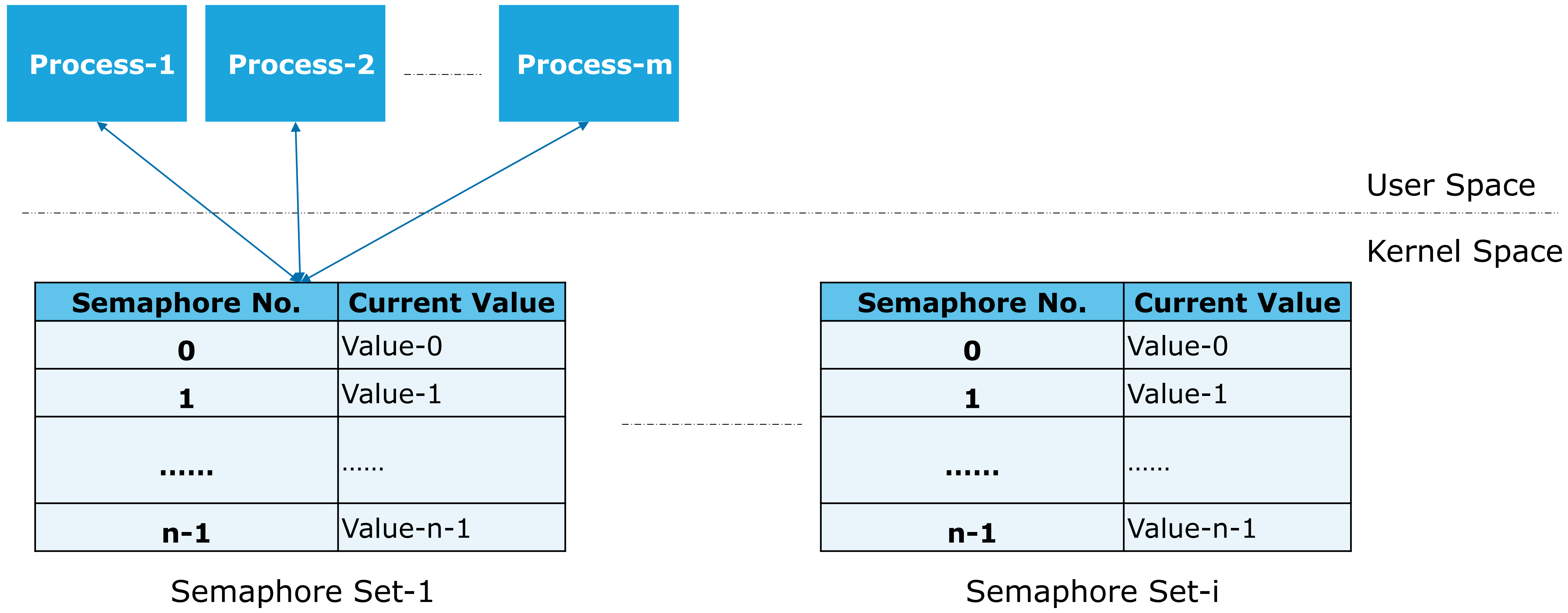
Semaphores

Introduction to System V Semaphore

- Synchronization object.
- Used to protect shared resources (Shared memory, Msg Queue etc).
- Set of Semaphores (Array of Semaphore).
- Upper limit of numbers of semaphores in a set: 25 (default).
- Operations on individual semaphore or set of semaphores.
- Operations based on Arithmatic.
- Any process with read permission can only test (read) the semaphore value.
- Any process with write permission can change, increment or decrement the semaphore value.
- When a process holding semaphore terminates without freeing the semaphore, it remains locked and waiting processes are deadlocked.
- Use SEM_UNDO option to avoid such a deadlock condition.

Semaphore Set & Process Synchronization



Semaphore No.	Current Value
0	Value-0
1	Value-1
……	……
n-1	Value-n-1

Semaphore No.	Current Value
0	Value-0
1	Value-1
……	……
n-1	Value-n-1

Semaphore Creation

- Set of multiple Semaphores.
- Maintained as array internally.
- Ownership & Permissions.
- int semget(key_t key, int nsems, int semflg);
- semget() creates and initializes a set of semaphores and returns a semaphore id for the set.
- Every process needing access to the semaphore set should call semget() to get semaphore id (semid) of the Semaphore set.
- Every process needing access to the semaphore set should be assigned separate access permission depending upon its usage.

Semaphore Control Operations

- int semctl(int semid, int nsems, int cmd, void *args)
- Used to control or query the status of a Semaphore set.
- Used to set/get values of a Semaphore or Semaphore set.
- Used to get the status of a Semaphore or Semaphore set.
- Used to change/set the properties (ownership, permissions etc) of a Semaphore or Semaphore set.
- Used to remove (delete) a Semaphore set.

Semaphore Control Operations (contd.)

Control Flags	Description
GETVAL	Return value of a single semaphore.
SETVAL	Set value of a single semaphore.
GETALL	Return value of all semaphores in a set.
SETALL	Set value of all semaphores in a set.
IPC_RMID	Remove (delete) the specified semaphore set.
IPC_STAT	Return the status information for the specified semaphore set.
IPC_SET	Set the effective user & group id and permission flags.
Misc.	

Semaphore Operations

- ➢ int semop(int semid, struct sembuf *sops, int nsops);
- ➢ semop() is by default a blocking system call.
- ➢ struct sembuf contains:
- • Semaphore number.
- • Operation to be performed.
- • Control flags.
- ➢ Operations Summary:
- • Positive integer value increments the semaphore values by that amount. Non-blocking.
- • Negative integer value decrements the semaphore values by that amount. Non-blocking.
- • Value of zero means to wait (block) for the semaphore value to reach zero.

Semaphore Operations (contd.)

- When an operation on a semaphore of a set fails, none of the semaphore values of the set are altered.
- Semaphore Operations control flags:
- IPC_NOWAIT – non-blocking call to semop().
- SEM_UNDO – system should release the semaphore set if process does not do it before termination.

Example

```
//Process-1
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define NUM_OF_SEM        2
main()
{
key_t key=123;
int nsems=NUM_OF_SEM;
int semflags=0666;
int semid;
struct sembuf sops[NUM_OF_SEM];
// Create Semaphore set of 2 Semaphores.
semid=semget(key, nsems, semflg); // TODO: Check the return value
// Set values of Semaphore set.
semctl(semid, 0, SETVAL, 1); // TODO: Check the return value
semctl(semid, 1, SETVAL, 1); // TODO: Check the return value
// TODO: Fill-up sops appropriately. Set values to 0 for waiting.
semop(semid, sops, NUM_OF_SEM); // TODO: Check the return value
// Access (read/write) the shared resource (shared memory etc) here
// Remove Semaphore.
semctl(semid, nsems, IPC_RMID, NULL);  // TODO: Check the return value
exit(0);
}
```

Example

```
//Process-2
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define NUM_OF_SEM          2
main()
{
key_t key=123;
int nsems=NUM_OF_SEM;
int semflags=0666;
int semid;
struct sembuf sops[NUM_OF_SEM];
// Create Semaphore set of 2 Semaphores.
semid=semget(key, nsems, semflg); //TODO: Check the return value
// Set values of Semaphore set.
// TODO: Fill-up sops appropriately. Set values to -1 for decrementing the semaphore value.
semop(semid, sops, NUM_OF_SEM); //TODO: Check the return value
// Remove Semaphore.
semctl(semid, nsems, IPC_RMID, NULL); //TODO: Check the return value
exit(0);
}
```