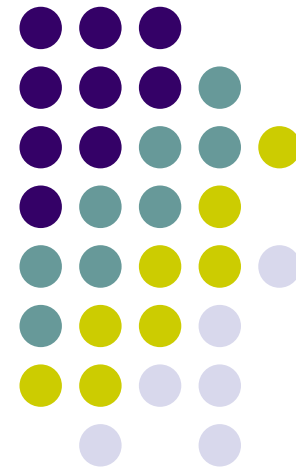


Einführung in die Systemprogrammierung

5

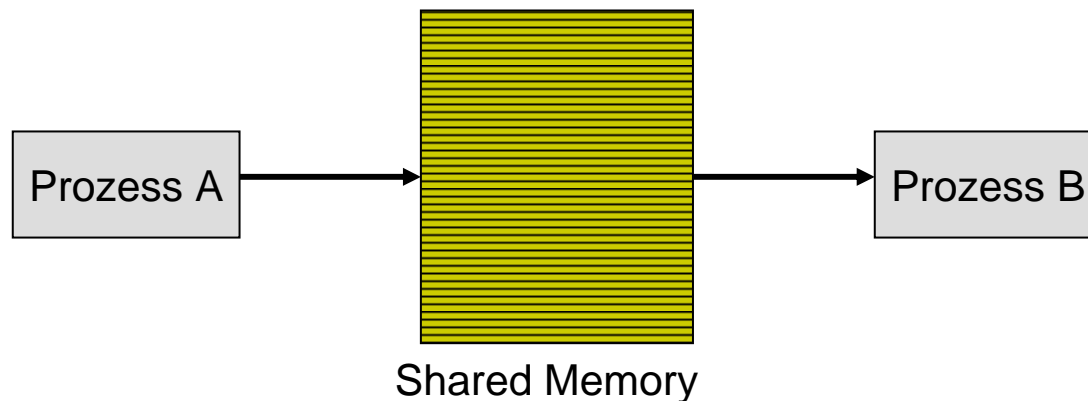
Interprozesskommunikation

Shared Memory



Shared Memory

- Zwei oder mehr Prozesse nutzen gemeinsam einen Speicherbereich
- Schnell, da Kopieren zwischen Speicherbereichen entfällt
- Zugriffe auf gemeinsamen Speicher muss synchronisiert werden!!!
 - verwenden von Semaphoren



shmid_ds – Status eines Shared Memory Segments

- Zu jedem Shared Memory Segment existiert eine shmid_ds-Struktur:

```
struct shmid_ds{
    struct ipc_perm sem_perm; // IDs und Zugriffsrechte
    int shm_segsz;           // Größe des Segments in Bytes
    ushort shm_lkcnt;       // Anzahl Sperren
    pid_t shm_lpid;         //PID des letzten shmop-Aufrufers
    pid_t shm_cpid;        //PID des Einrichters des SM
    ulong shm_nattach;      //Anzahl angebundener Prozesse
    time_t shm_atime;       //Letzter attach-Zeitpunkt
    time_t shm_dtime;       //Letzter detach-Zeitpunkt
    time_t sem_ctime;       //Zeit d. letzten Änderung
}
```

shmget – Öffnen oder Kreieren eines Shared Memory Segments

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

int shmget (key_t schlüssel, int groesse, int flag);
```

- Rückgabewerte:
 - Bei Erfolg: Kennung des Shared Memory Segments
 - Bei Fehler: -1
- *groesse*: Minimale Größe des Shared Memory Segments
 - Beim Öffnen existierender Segmente: 0
- *flag*:
 - IPC_CREAT: zum kreieren eines neuen Objekts mit dem angegebenen Schlüssel

shmctl –Löschen eines Segments

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

int shmctl (int kennung, int kdo, struct shmid_ds *puffer);
```

- Rückgabewerte:
 - Bei Erfolg: 0
 - Bei Fehler: -1
- *kdo*: legt die durchzuführende Aktion fest:
 - IPC_RMID: Löschen des Segments
- **Hinweis: Shared Memory Segmente werden erst dann wirklich gelöscht, wenn kein Prozess mehr an dieses Segment angebunden ist (`shm_nattach==0`). Alle Prozesse müssen beendet sein oder die Anbindung aufgehoben haben**

shmat – Anbinden eines Segments

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

void *shmat (int kennung, void *adr, int flag);
```

- Rückgabewerte:
 - Bei Erfolg: Adresse des Shared Memory Segments
 - Bei Fehler: -1
- `adr == NULL`: Kernel legt Adresse des Shared Memory-Segments fest
- `flag`: Diverse Optionen
 - 0: keine
 - `SHM_RDONLY`: Segment nur zum lesen angebunden

shmdt – Loslösen eines Segments

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

void *shmdt (void *adr);
```

- Rückgabewerte:
 - Bei Erfolg: 0
 - Bei Fehler: -1
- adr: Adresse des Shared Memory Segments
 - Rückgabewert eines vorangegangenen shmat-Aufrufs

Beispiel – Headerdatei

```
#ifndef SM
#define SM

#define SHM_MYKEY1 10008
#define MAX_ANTWORT 100
#define SHM_MAXSAETZE 1000
// Definition der Datensatzstruktur
typedef struct {
    pid_t pid;
    char ergebnis[MAX_ANTWORT];
    char ungelesen;
}shm_satz;
// Definition der Shared Memory-Struktur und Größe
typedef struct{
    long satznr;
    shm_satz antwort[SHM_MAXSAETZE];
}antwort_shm;

#endif
```

Beispiel – Einrichten und Anbinden

```
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include "sm.h"

static int shm_antwortid;

int main(void)
{
    int schreib_satznr, lese_satznr;
    antwort_shm *shm_antwort;
    pid_t pid;
    char ergebnis[MAX_ANTWORT];

    /* Einrichten und Anbinden des Shared Memory */
    if ( (shm_antwortid=shmget(SHM_MYKEY1,
        SHM_MAXSAETZE*MAX_ANTWORT+1, 0766|IPC_CREAT|IPC_EXCL) )== -1)
        // Fehlerbehandlung;

    if ( ( shm_antwort=
        (antwort_shm *)shmat(shm_antwortid, NULL, 0))==(void *)-1)
        // Fehlerbehandlung ;
```

Beispiel – Schreiben

```
/* Initialisierung */
shm_antwort->satznr=-1;      // noch keine Sätze vorhanden
schreib_satznr=0;
lese_satznr=0;

/*Schreiben in Shared Memory*/
schreib_satznr=shm_antwort->satznr;
schreib_satznr = ++schreib_satznr % SHM_MAXSAETZE;
shm_antwort->antwort[schreib_satznr].pid=getpid();
strcpy(
    shm_antwort->antwort[schreib_satznr].ergebnis,"Hallo\n");

shm_antwort->antwort[schreib_satznr].ungelesen=1;
shm_antwort->satznr=schreib_satznr;

printf("Datensatz geschrieben: satznr= %i, pid= %i \n",
    schreib_satznr, getpid());
```

Beispiel – Lesen und Löschen

```
/* Lesen aus Shared Memory */
if (shm_antwort->antwort[lese_satznr].ungelesen==1)
{
    shm_antwort->antwort[lese_satznr].ungelesen=0;
    pid_t rdpid=shm_antwort->antwort[lese_satznr].pid;
    strcpy(
        ergebnis,shm_antwort->antwort[schreib_satznr].ergebnis);
    printf("Datensatz gelesen: satznr= %i, pid= %i \n",
        lese_satznr, rdpid);
    lese_satznr = ++lese_satznr%SHM_MAXSAETZE;
}

/* Löschen des Shared Memory */
if (shmctl(shm_antwortid, IPC_RMID, NULL)==-1)
    //Fehlerbehandlung ;
}
```