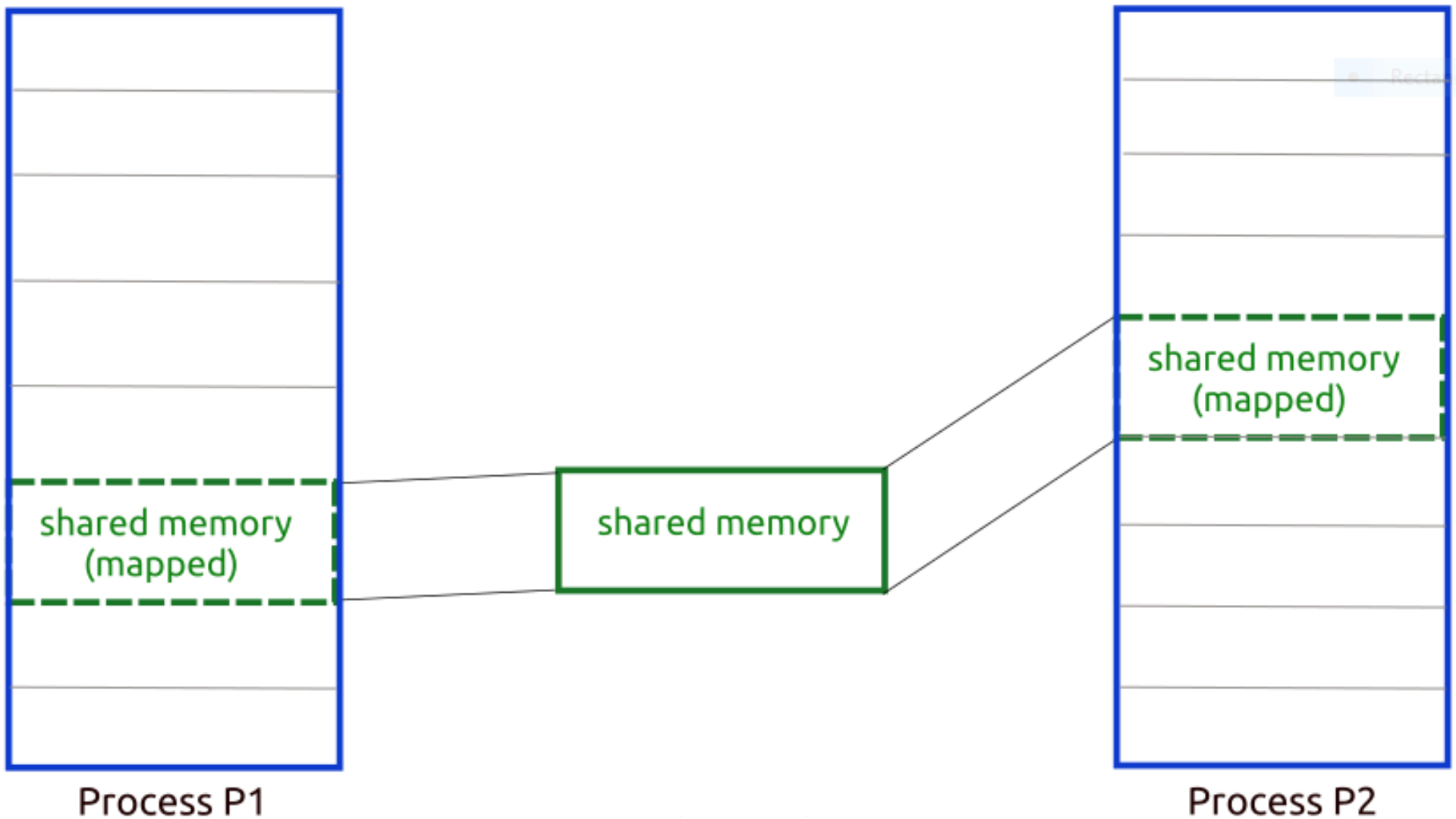
shared memory
(mapped)
shared memory
shared memory
(mapped)
Process P1
Process P2

Write a program to get a string from the keyboard and write into memory. Also write another program to read from the memory.

P1.C-WRITING TO MEMORY

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>
Const int SHMSZ=1000;
Main()
{
Int sid;
Key_t key;
Char *sbuff;
If((key = ftok("/desk/abc",'a'))==-1)
{  printf("ftok error");}
/*shared memory id*/
If((sid=shmget(key,SHMSZ,0644/IPC_CREAT))==-1)
{Printf("shmget error");
}
```

```
If((sbuff= (char*)shmat (sid, NULL, SHM_RDONLY)) == (char*) -1)
{Printf("shmat error");
}
Memcpy(sbuff,"HELLO",5);
Sleep(30);
}
```

P2.C-READING TO MEMORY

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>
Const int SHMSZ=1000;
Main()
{
Int sid;
Key_t key;
Char *sbuff;
/*shared memory id*/
If((sid=shmget(key,SHMSZ,0))==-1)
{Printf("shmget error");
}
```

```
#inclue<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>
Const int SHMSZ=1000;
Main()
{
Int sid;
Key_t key;
Char *sbuff;
/*shared memory id*/
If((sid=shmget(key,SHMSZ,0))==-1)
{Printf("shmget error");
}
```

```
If((sbuff= (char*)shmat (sid, NULL, 0)) == (VOID*) -1)
{Printf("shmat error");
}
PRINTF("MESSAGE FROM P1 IS =%s", sbuff);
}
```

2. **Program to write a-z using shared memory**

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>
Const int SHMSZ=100;
Main()
{
Char c;
Int shmid;
Key_t key;
Char *shm, *s;
/*shared memory id*/
If((shmid=shmget(4678,SHMSZ, IPC_CREAT|0666))
< 0)
{Printf("shmget error");
}
```

```
If((shm= shmat (shmid, NULL, 0)) == (char*) -1)
{Printf("shmat error");
}
s=shm;
For (c='a';c<='z';c++)
{*s++=c;
}
*s='\0';
}
```

Program to read from shared memory

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>
Const int SHMSZ=100;
Main()
{
Char c;
Int shmid;
Key_t key;
Char *shm, *s;
/*shared memory id*/
If((shmid=shmget(4678,SHMSZ, 0)) = = -1)
{Printf("shmget error");
}
If((shm=shmat(shmid,NULL, 0)) = = (char*)-1)
{Printf("shmat error");
}
Printf("message read is %s",shm);
}
Shmdt(shm);
}
```

Semaphores-Producer consumer (writing/reading from 1 to 50)

```
#include<sys/sem.h>
#define P(S) semop(S,&pop,1)
#define V(S) semop(S,&vop,1)
Main()
{int *a,*b;
Int I,j,count=50;status;
Int semid1,semid2;
Struct sembuf pop,vop;
Semid1=semget(IPC_PRIVATE,1,0777|IPC_CREAT);
Semid2=semget(IPC_PRIVATE,1,0777|IPC_CREAT);
Semctl(semid1,0,SETVAL,0);
Semctl(semid2,0,SETVAL,1);
Pop.sem_num=vop.sem_num=0;
Pop.sem_flg=vop.sem_flg=0;
Pop-sem_op=-1;vop.sem_op=1;
If(fork()==0)
{ /*child:consumer*/
FILE *fp;
Int data;
While(count)
{
P(semid1);
Fp=fopen("datafile",'r");
```

```
Fscanf(fp,"%d",&data);
Printf("consumer reads
%d\n",data);
Fclose(fp);
V(semid2);
Count--;
}
}
Else{/*parent:producer*/
FILE *fp;
Int data=0;
While(count)
{sleep(1);
P(semid2);
Fp=fopen("datafile","w");
Fprintf(fp,"%d\n",data);
Data++;
Fclose(fp);
V(semid1);
Count--;
}
```

```
Wait(&status);
Semctl(semid1,0,IPC_RMID,0);
Semctl(semid2,0,IPC_RMID,0);
}
}
```