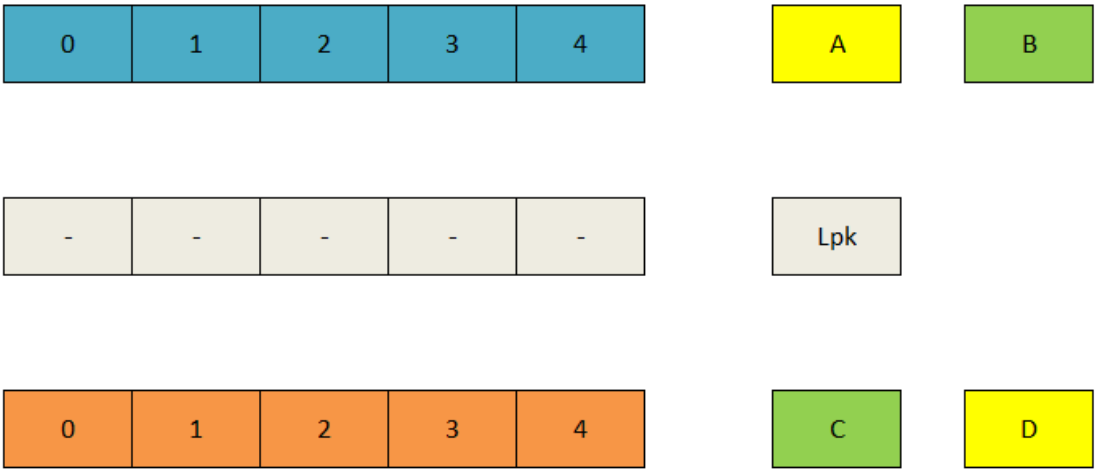
Szymon Krysztopolski
144619
I4.1

Sytuacja początkowa:
(bufor główny oraz tablice pomocnicze w programie mają rozmiar 10)



Napisane zostały 3 programy: *producent*, *konsument*, oraz *start*. Program *start* służy do wyzerowania ustawień do tych początkowych, tj.: podniesienie lub opuszczenie odpowiednich semaforów do konkretnych wartości, wyzerowanie zmiennych współdzielonych, ustawienie tablic pomocniczych do ustawień początkowych.

Wyjaśnienie znaczenia zmiennych:
Tablica pomocnicza 1 (niebieska) – ws[Rozmiar] – tablica przechowująca kolejne wolne sloty
Tablica pomocnicza 2 (pomarańczowa) – prod[Rozmiar] – tablica przechowująca produkty
Powyższe tablice przechowują index'y do głównego bufora

Zmienna A – następny index w tablicy ws[] który można zwolnić
Zmienna B – następny index w tablicy ws[] na którym można produkować
Zmienna C – następny index w tablicy prod[] na który można umieścić wyprodukowany produkt
Zmienna D – następny index w tablicy prod[] z którego można pobrać produkt

Zmienne zaznaczone na żółto (A,D) są używane tylko przez konsumentów.
Zmienne zaznaczone na zielono (B,C) są używane tylko przez producentów.

Aby pokazać konsumentom że warto czekać na produkty, użyto dodatkowo zmienną *Lpk* (liczba potencjalnych produktów). W momencie gdy zmienna jest dodatnia konsument zmniejsza ją o jeden informując innych konsumentów, że to on zajmie się jednym zadaniem.

Wszelkie operacje wykonywane na powyższych zmiennych/tablicach są wykonywane w sekcji krytycznej S_bin (jest to semafor binarny).

Tablica zaznaczona na szaro symbolizuje główny bufor, do którego procesy mają dostęp współbieżny.

Przedrostek *sk* przy definiowaniu obszarów pamięci współdzielonej pochodzi od inicjałów twórcy.

PROGRAM START

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <semaphore.h>

#define Rozmiar 10 //rozmiar tablicy gl buf
#define BUF_SIZE 4096

int main()
{
	sem_t *S_bin,*Sk,*Sp;
	int e;
	int fd_Lpk;//liczba potencjalnych produktow
	int *Lpk;
	int fd_A,fd_B,fd_C,fd_D;
	int *A,*B,*C,*D; //znacznie opisane w instrukcji
	int fd_ws,fd_prod,fd_buff;
	int *ws,*prod,*buff;

	S_bin=sem_open("/sk_S_bin",O_CREAT,0600,1);
	Sk=sem_open("/sk_konsumenci",O_CREAT,0600,0);
	Sp=sem_open("/sk_producenci",O_CREAT,0600,Rozmiar);

	fd_Lpk=shm_open("/sk_Lpk",O_CREAT|O_RDWR,0600);
	fd_A=shm_open("/sk_A",O_CREAT|O_RDWR,0600);
	fd_B=shm_open("/sk_B",O_CREAT|O_RDWR,0600);
	fd_C=shm_open("/sk_C",O_CREAT|O_RDWR,0600);
	fd_D=shm_open("/sk_D",O_CREAT|O_RDWR,0600);
	fd_ws=shm_open("/sk_ws",O_CREAT|O_RDWR,0600);//wole sloty
	fd_prod=shm_open("/sk_prod",O_CREAT|O_RDWR,0600);//produkty
	fd_buff=shm_open("/sk_buff",O_CREAT|O_RDWR,0600);//buff glowny

	ftruncate(fd_Lpk,sizeof(int));
	ftruncate(fd_A,sizeof(int));
	ftruncate(fd_B,sizeof(int));
	ftruncate(fd_C,sizeof(int));
	ftruncate(fd_D,sizeof(int));
	ftruncate(fd_ws,Rozmiar*sizeof(int));
	ftruncate(fd_prod,Rozmiar*sizeof(int));
	ftruncate(fd_buff,Rozmiar*sizeof(int));
	Lpk=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_Lpk,0);
	A=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_A,0);
	B=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_B,0);
	C=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_C,0);
	D=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_D,0);
	ws=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_ws,0);
	prod=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_prod,0);
	buff=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_buff,0);
```

```
	if(Lpk==MAP_FAILED)e=1;
	if(A==MAP_FAILED)e=1;
	if(B==MAP_FAILED)e=1;
	if(C==MAP_FAILED)e=1;
	if(D==MAP_FAILED)e=1;
	if(S_bin==SEM_FAILED)e=2;
	if(Sk==SEM_FAILED)e=2;
	if(Sp==SEM_FAILED)e=2;

	if(e==1){
		perror("mmap2");
		exit(1);
	}
	if(e==2){
		perror("sem");
		exit(2);
	}

	*A=0;
	*B=0;
	*C=0;
	*D=0;
	for(int i=0;i<Rozmiar;i++) ws[i]=i;
	for(int i=0;i<Rozmiar;i++) prod[i]=i;
	*Lpk=0;
	//printf("Lpk %d\n",*Lpk);

	int tmp,tmp1,tmp2;
	sem_wait(S_bin);
	sem_getvalue(Sk,&tmp);while(tmp>0){sem_wait(Sk);sem_getvalue(Sk,&tmp);}
	sem_getvalue(Sp,&tmp1);while(tmp1<Rozmiar){sem_post(Sp);sem_getvalue(Sp,&tmp1);}
	sem_getvalue(S_bin,&tmp2);
	printf("Sk %d Sp %d S_bin %d\n",tmp,tmp1,tmp2);
	sem_post(S_bin);

	sem_close(S_bin);
	sem_close(Sk);
	sem_close(Sp);
	sem_unlink("/S_bin");
	sem_unlink("/Sk");
	sem_unlink("/Sp");

	return 0;
}
```

PROGRAM PRODUCENT

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <time.h>

#define Rozmiar 10 //rozmiar tablicy gl buf
#define BUF_SIZE 4096

int main(int ile,char **arg)
{
	srand(time(NULL));
	int N=atoi(arg[1]);
	sem_t *S_bin,*Sk,*Sp;
	int pid,e=0;
	int fd_Lpk;//liczba potencjalnych produktow
	int *Lpk;
	int fd_A,fd_B,fd_C,fd_D;
	int *A,*B,*C,*D; //znacznie opisane w instrukcji
	int fd_ws,fd_prod,fd_buff;
	int *ws,*prod,*buff;

	S_bin=sem_open("/sk_S_bin",O_CREAT,0600,1);
	Sk=sem_open("/sk_konsumenci",O_CREAT,0600,0);
	Sp=sem_open("/sk_producenci",O_CREAT,0600,Rozmiar);

	fd_Lpk=shm_open("/sk_Lpk",O_CREAT|O_RDWR,0600);
	fd_A=shm_open("/sk_A",O_CREAT|O_RDWR,0600);
	fd_B=shm_open("/sk_B",O_CREAT|O_RDWR,0600);
	fd_C=shm_open("/sk_C",O_CREAT|O_RDWR,0600);
	fd_D=shm_open("/sk_D",O_CREAT|O_RDWR,0600);
	fd_ws=shm_open("/sk_ws",O_CREAT|O_RDWR,0600);//wole sloty
	fd_prod=shm_open("/sk_prod",O_CREAT|O_RDWR,0600);//produkty
	fd_buff=shm_open("/sk_buff",O_CREAT|O_RDWR,0600);//buff glowny

	ftruncate(fd_Lpk,sizeof(int));
	ftruncate(fd_A,sizeof(int));
	ftruncate(fd_B,sizeof(int));
	ftruncate(fd_C,sizeof(int));
	ftruncate(fd_D,sizeof(int));
	ftruncate(fd_ws,Rozmiar*sizeof(int));
	ftruncate(fd_prod,Rozmiar*sizeof(int));
	ftruncate(fd_buff,Rozmiar*sizeof(int));
	Lpk=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_Lpk,0);
	A=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_A,0);
	B=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_B,0);
	C=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_C,0);
	D=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_D,0);
	ws=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_ws,0);
```

```
	prod=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_prod,0);
	buff=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_buff,0);

	if(Lpk==MAP_FAILED)e=1;
	if(A==MAP_FAILED)e=1;
	if(B==MAP_FAILED)e=1;
	if(C==MAP_FAILED)e=1;
	if(D==MAP_FAILED)e=1;
	if(S_bin==SEM_FAILED)e=2;
	if(Sk==SEM_FAILED)e=2;
	if(Sp==SEM_FAILED)e=2;
	if(ile!=2)e=3;
	if(N<1)e=3;

	if(e==1){
		perror("mmap2");
		exit(1);
	}
	else if(e==2){
		perror("sem_open");
		exit(2);
	}
	else if(e==3){
		perror("argumenty");
		exit(3);
	}

	pid=getpid();
	sem_wait(S_bin);
	*Lpk+=N;
	sem_post(S_bin);
	int tmp,pobraneWS,oddanePD;
	while(N)
	{
		printf("Producent %d czeka\n",pid);
		sem_wait(Sp);
		sem_wait(S_bin);

		pobraneWS=prod[*B];
		*B+=1;
		*B%=Rozmiar;
		sem_post(S_bin);

		tmp=rand()%200+10;
		buff[pobraneWS]=tmp;
		printf("Producent %d jest aktywny i zapisuje wartosc %d na miejsce
%d\n(pracuje)\n",pid,tmp,pobraneWS);
		sleep(rand()%10+2);

		sem_wait(S_bin);
		oddanePD=*C;
		*C+=1;
```

```
            *C%=Rozmiar;
            prod[oddanePD]=pobraneWS;
            sem_post(S_bin);
            sem_post(Sk);
            printf("Producent %d konczy prace i zapelnia slot nr %d\n",pid,oddanePD);
            N-=1;
        }
        printf("Producent %d nie ma nic do zrobienia\n",pid);
        sem_close(S_bin);
        sem_close(Sk);
        sem_close(Sp);
        sem_unlink("/S_bin");
        sem_unlink("/Sk");
        sem_unlink("/Sp");
        return 0;
}
```

PROGRAM KONSUMENT

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <time.h>

#define Rozmiar 10 //rozmiar tablicy gl buf
#define BUF_SIZE 4096

int main()
{
	srand(time(NULL));
	sem_t *S_bin,*Sk,*Sp;
	int pid,e=0;
	int fd_Lpk;//liczba potencjalnych produktow
	int *Lpk;
	int fd_A,fd_B,fd_C,fd_D;
	int *A,*B,*C,*D; //znacznie opisane w instrukcji
	int fd_ws,fd_prod,fd_buff;
	int *ws,*prod,*buff;

	S_bin=sem_open("/sk_S_bin",O_CREAT,0600,1);
	Sk=sem_open("/sk_konsumenci",O_CREAT,0600,0);
	Sp=sem_open("/sk_producenci",O_CREAT,0600,Rozmiar);

	fd_Lpk=shm_open("/sk_Lpk",O_CREAT|O_RDWR,0600);
	fd_A=shm_open("/sk_A",O_CREAT|O_RDWR,0600);
	fd_B=shm_open("/sk_B",O_CREAT|O_RDWR,0600);
	fd_C=shm_open("/sk_C",O_CREAT|O_RDWR,0600);
	fd_D=shm_open("/sk_D",O_CREAT|O_RDWR,0600);
	fd_ws=shm_open("/sk_ws",O_CREAT|O_RDWR,0600);//wole sloty
	fd_prod=shm_open("/sk_prod",O_CREAT|O_RDWR,0600);//produkty
	fd_buff=shm_open("/sk_buff",O_CREAT|O_RDWR,0600);//buff glowny

	ftruncate(fd_Lpk,sizeof(int));
	ftruncate(fd_A,sizeof(int));
	ftruncate(fd_B,sizeof(int));
	ftruncate(fd_C,sizeof(int));
	ftruncate(fd_D,sizeof(int));
	ftruncate(fd_ws,Rozmiar*sizeof(int));
	ftruncate(fd_prod,Rozmiar*sizeof(int));
	ftruncate(fd_buff,Rozmiar*sizeof(int));
	Lpk=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_Lpk,0);
	A=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_A,0);
	B=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_B,0);
	C=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_C,0);
	D=mmap(NULL,sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_D,0);
	ws=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_ws,0);
	prod=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_prod,0);
```

```
	buff=mmap(NULL,Rozmiar*sizeof(int),PROT_READ|PROT_WRITE,MAP_SHARED,fd_buff,0);

	if(Lpk==MAP_FAILED)e=1;
	if(A==MAP_FAILED)e=1;
	if(B==MAP_FAILED)e=1;
	if(C==MAP_FAILED)e=1;
	if(D==MAP_FAILED)e=1;
	if(S_bin==SEM_FAILED)e=2;
	if(Sk==SEM_FAILED)e=2;
	if(Sp==SEM_FAILED)e=2;

	if(e==1){
		perror("mmap2");
		exit(1);
	}
	else if(e==2){
		perror("sem_open");
		exit(2);
	}

	pid=getpid();
	int aktywny;
	sem_wait(S_bin);
	if(*Lpk>0){
		aktywny=1;
		*Lpk-=1;
	}
	else aktywny=0;
	sem_post(S_bin);
	int tmp,pobranePD,oddaneWS;
	while(aktywny)
	{
		printf("Konsument %d czeka\n",pid);
		sem_wait(Sk);
		sem_wait(S_bin);
		pobranePD=prod[*D];
		*D+=1;
		*D%=Rozmiar;
		sem_post(S_bin);
		tmp=buff[pobranePD];

		printf("Konsument %d jest aktywny i odczytuje wartosc %d z miejsca
%d\n(pracuje)\n",pid,tmp,pobranePD);
		sleep(rand()%10+2);

		sem_wait(S_bin);
		oddaneWS=*A;
		*A+=1;
		*A%=Rozmiar;
		prod[oddaneWS]=pobranePD;
		sem_post(S_bin);
		sem_post(Sp);
```

```
            printf("Konsument %d konczy prace i zwalnia slot nr %d\n",pid,oddaneWS);

            sem_wait(S_bin);
            if(*Lpk<=0)aktywny=0;
            else *Lpk-=1;
            sem_post(S_bin);
        }
        printf("Konsument %d nie ma nic do zrobienia\n",pid);
        sem_close(S_bin);
        sem_close(Sk);
        sem_close(Sp);
        sem_unlink("/S_bin");
        sem_unlink("/Sk");
        sem_unlink("/Sp");
        return 0;
}
```