Relatorio Trabalho de semaforos 03/10/2016
Sistemas Operacionais – GBC045
Ciencia da Computacao
Rivalino Matias

Gabriel Augusto Ferraz Martins
11221BCC036

Descricao do PC
Processador: Core 2 Duo T5800 / 2 GHz - Centrino
Memoria Principal : 4 GB
Sistema Operacional : Linux parrot
Kernel : 4.6.0-parrot-amd64 #1 SMP Parrot 4.6.3-1parrot1 (2016-07-15)

Descricao do trabalho
Verificar as diferencas entre implementacao e resultado mediante as diferentes abordagens de implementacao de um servico consumidor e cliente utilizando formas distintas de comunicacao entre processos. Que no caso especifico, estara tratando um c'odigo que utiliza de "busy-wait", para eventual implementacao em "semaforos"(os dois utilizando IPC de Shared-Memory)

Resultados
Para se utilizar a comunicacao entre processos de semaforos, foi necessario a criacao de uma nova variavel do tipo semaforo dentro da estrutura que estava sendo compartilhada entre os processos na 'area de memoria em comum. Esta variavel ira ser respons'avel por inicializar, e fazer o manejamento das outras operacoes do semaforo.

Antes da implementacao do semaforo, podemos observar que tanto o consumidor, quando o produtor, aguardavam a producao ou o consumo de outro processo da 'area de memoria compartilhada. Quando os dois sao executados em conjunto, ambos utilizam 100% da CPU no momento de execucao, ate serem interrompidos.

Depois da implementacao do semaforo, podemos observar que tanto o consumidor quanto o produtor, entravam automaticamente no laco infinitamente, sem aguardar que um novo processo utilize a 'area de memoria compartilhada. Por'em, assim que os dois eram executados em conjunto, podiamos observar que a utilizacao da CPU nao ficava em 100%, o produtor utilizava em entre 40 e 80% de CPU, enquanto o consumidor ficava na faixa de 6-12%

Secao de codigos – Produtor

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <semaphore.h>

#define MEM_SZ 4096
#define BUFF_SZ MEM_SZ-sizeof(int)

struct shared_area{
	char buffer[BUFF_SZ];
	sem_t mutexshared;
}

main()
{
	int i;
	key_t key=1234;
	struct shared_area *shared_area_ptr;
	void *shared_memory = (void *)0;
	int shmid;

	shmid = shmget(key,MEM_SZ,0666|IPC_CREAT);
	if ( shmid == -1 )
	{
		printf("shmget falhou\n");
		exit(-1);
	}

	printf("shmid=%d\n",shmid);

	shared_memory = shmat(shmid,(void*)0,0);

	if (shared_memory == (void *) -1 )
	{
		printf("shmat falhou\n");
		exit(-1);
	}

	printf("Memoria compartilhada no endereco=%x\n",(int) shared_memory);

	shared_area_ptr = (struct shared_area *) shared_memory;
	sem_init(&(shared_area_ptr->mutexshared),1,1);

	for(i=0;i<BUFF_SZ;i++)
		shared_area_ptr->buffer[i]=0;

	for(;;)
	{
		sem_wait(&(shared_area_ptr->mutexshared));
```

```
                for(i=0;i<BUFF_SZ;i++)
                        shared_area_ptr->buffer[i]='#';
                sem_post(&(shared_area_ptr->mutexshared));
                printf("Produziu %d bytes\n",i);
        }
        sem_destroy(&(shared_area_ptr->mutexshared));
    exit(0);
}
```

Secao de codigos – Consumidor

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <semaphore.h>

#define MEM_SZ 4096
#define BUFF_SZ MEM_SZ-sizeof(int)

struct shared_area{
	char buffer[BUFF_SZ];
	sem_t mutex
}


main()
{
	int i;
	key_t key=1234;
	struct shared_area *shared_area_ptr;
	void *shared_memory = (void *)0;
	int shmid;

	shmid = shmget(key,MEM_SZ,0666|IPC_CREAT);
	if ( shmid == -1 )
	{
		printf("shmget falhou\n");
		exit(-1);
	}

	printf("shmid=%d\n",shmid);

	shared_memory = shmat(shmid,(void*)0,0);

	if (shared_memory == (void *) -1 )
	{
		printf("shmat falhou\n");
		exit(-1);
	}

	printf("Memoria compartilhada no endereco=%x\n",(int) shared_memory);

	shared_area_ptr = (struct shared_area *) shared_memory;
	sem_init(&(shared_area_ptr->mutex),1,1);

	for(i=0;i<BUFF_SZ;i++)
		shared_area_ptr->buffer[i]=0;

	for(;;)
	{
```

```
                        sem_wait(&(shared_area_ptr->mutex));
                        for(i=0;i<BUFF_SZ;i++){
                                printf("%c",shared_area_ptr->buffer[i]);
                        }
                        printf("\nConsumiu %d bytes\n",i);
                        sem_post(&(shared_area_ptr->mutex));
        }
        sem_destroy(&(shared_area_ptr->mutex));
    exit(0);
}
```