PID -> identificador do processo

Funções para processos filhos – Experimento 1

Criar processo: fork() -> criar processo filho
Retorna 0 no processo filho e um número diferente de 0 no pai (PID do filho)

Matar processo: kill(id do processo que deseja matar, sinal (geralmente 1))

Troca o código do processo: execl(nome, nome, parâmetros, (char *) NULL)
Passar o nome do novo código das vezes, os parâmetros numa string e NULL no final, tem várias outras funções da família exec

Funções de filas de mensagens – Experimento 2

Deve se definir a estrutura:

```
typedef struct  {
long mtype;
char mtext[tamanho];
} msgbuf_t;
```

mtext vai ter os dados que se deseja enviar, mtype vai conter o tipo da mensagem.
fazer esqueminha do dowscasting

```
 Tipostruct *aux = (Tipostuct *) (mensagem.mtext);
```

Cria fila de mensagens: msgget(chave, IPC_CREAT | 0666)
Passar uma chave única e a flag de acesso (IPC_CREAT | 0666) e retorna o identificador da fila

Enviar dados a fila: msgsnd(id da fila, ponteiro para os dados, tamanho dos dados, flag (0)) *fazer cast ponteiro (struct msgbuf *)

Receber dados da fila: msgrcv(id da fila, ponteiro para aonde os dados vai ficar, tipo da mensagem, flag (0)) *fazer cast ponteiro (struct msgbuf *)

Destruir fila: msgctl(id da fila, IPC_RMID,NULL)

Funções de memória compartilhada – Experimento 3

Cria segmento de memória: shmget(chave, tamanho, IPC_CREAT | 0666)
Retorna o id do segmento de memoria

Associar segmento de memória: (int *)shmat(id do segmento, NULL, 0) *fazer cast do int
Retorna o endereço do segmento de memoria

Destruir segmento de memória: shmctl(id do segmento, IPC_RMID,NULL)

Funções de semáforo – Experimento 3

Estrutura do semáforo
g_sem_op1[0].sem_num = 0;
g_sem_op1[0].sem_op = -1 para trancar e 1 pra destrancar
g_sem_op1[0].sem_flg = 0;

Criar semáforo: semget (chave do semáforo, número de posições do vetor que ira contar, IPC_CREAT | 0666)
Retorna o identificador do semáforo

Travar semáforo: semop(id, estrutura pra travar, 1)

Destravar semáforo: semop(id, estrutura pra destravar,1)

Destruir semáforo: semctl(id, IPC_RMID, 0)

Semáforo Mutex – Experimento 4

Estrutura: pthread_mutex_t nomeDoSemaforo;

Inicializar: pthread_mutex_init(&(nome), NULL);

Travar: pthread_mutex_lock(&(nome));

Destravar: pthread_mutex_unlock(&(nome));

Threads – Experimento 4

Vetor com id das threads: pthread_t vetor[qtd];

Criar threads: pthread_create(&vetor[i], NULL, procedimento, (void *) argumento);

Esperar thread: pthread_join(vetor[i], NULL);

Terminar mas manter as outras ainda: pthread_exit(vetor[i], NULL);

Matar todas: exit(0);

Produtor x Consumidor

Semáforos ler=0, escrever=N

Produtor	Consumidor
Trava ler Le Destrava escrever	Trava escrever Escreve Destrava ler