Universidade de Brasília - Instituto de Ciências Exatas

Departamento de Ciência da Computação

Programação Concorrente - Trabalho 1:

Processos sincronizados em empresa de entregas.

Aluno: Gustavo Rinaldi Braga de Albuquerque

Matrícula: 222008664

Brasília, DF

2025

Introdução:

O objetivo da execução do trabalho era criar um programa em C que utilizasse de processos concorrentes para a solução de um problema. Tendo isso em vista, utilizei da biblioteca POSIX Pthreads em C para criar uma empresa de entregas que possuía processos simultâneos entre produção e entregas de produtos.

Descrição do tema:

O problema é gerado por uma empresa fictícia que produz produtos sobre demanda, sendo assim é necessário que os pedidos sejam buscados para assim começar a produção, e depois de finalizados, poderão ser entregues. Um dos obstáculos dessa empresa é que os entregadores só possuem uma chave de acesso a fábrica, então todos devem estar ou buscando os pedidos(cartas) ou levando os produtos já empacotados para seus compradores. Outro problema é que essa empresa não possuí um grande estoque, então é necessário que caso haja produtos finalizados, eles sejam entregues preferencialmente.

Na parte da produção e empacotamento da empresa, há uma dependência em relação aos pedidos(cartas), já que precisam deles para começar a produção, e posteriormente, o empacotamento. Os empacotadores só poderão iniciar seus trabalhos caso a produção seja finalizada, e após o empacotamento os produtos estarão prontos para serem entregues.

Descrição técnica do programa:

Primeiramente explicarei cada um dos processos envolvidos do programa, sendo eles: entregador_cartas, entregador_entregas, producao e empacotamento. Após apresentar os processos, abordarei os lock e variáveis de condição de forma mais detalhada.

O processo entregador_cartas ele é o primeiro passo para o funcionamento do projeto, pois sem ele a produção não pode ser iniciada. O entregador_cartas verifica inicialmente se há entregas pendentes e se outro processo de entrega já está em andamento, caso encontre um processo ativo, ele aguardará o sinal para começar a buscar cartas. Quando estiver disponível, ele adquirirá o lock de entregas, busca as cartas e sinaliza o processo de produção.

```
pthread_mutex_lock(&mutex_ent);
while (entregando != 0 || empacotamento_concluido != 0) {
    pthread_cond_wait(&cond_buscar, &mutex_ent);
}
buscando++;
id_c++;
printf("Entregador %d: Buscando carta %d...\n", id, id_c);
```


O processo produção ele espera a busca de cartas para ser acordado, para assim começar a produção e adquirir o lock de produção e poder transferir o produto para o próximo estágio,

acordando assim o processo do empacotamento. Caso não exista pedidos, ele printará que está a espera de mais pedidos.

```c
while (1) {
    // Inicia a produção quando há pedidos para produzir.
    pthread_mutex_lock(&mutex_ent);
    while (pedidos_buscados == 0) {
        printf("Produtor %d: Aguardando pedidos...\n", id);
        pthread_cond_wait(&cond_produzir, &mutex_ent);
    }
    pedidos_buscados--;
    pthread_mutex_unlock(&mutex_ent);

    sleep(1);
    // Quando finaliza  produção ele inicia o empacotamento.
    pthread_mutex_lock(&mutex_prod);
    id_p++;
    printf("Produtor %d: Produzindo pedido %d...\n", id, id_p);
    producao_concluida++;
    pthread_cond_signal(&cond_empacotar);
    pthread_mutex_unlock(&mutex_prod);
```

O empacotamento é iniciado após a conclusão da produção. Este processo empacota os produtos e sinaliza o entregador_entregas para que as entregas possam ser realizadas. Durante este estágio, é utilizado o lock de entrega para garantir que a variável de controle seja manipulada sem inconsistências.

```c
while (1) {
    // Inicia o empacotamento depois da produção desse produto.
    pthread_mutex_lock(&mutex_prod);
    while (producao_concluida == 0) {
        printf("Empacotador %d: Aguardando produção...\n", id);
        pthread_cond_wait(&cond_empacotar, &mutex_prod);
    }
    producao_concluida--;
    pthread_mutex_unlock(&mutex_prod);

    // Quando finaliza o empacotamento, acorda o entregador para ser entregue.
    pthread_mutex_lock(&mutex_ent);
    id_ep++;
    printf("Empacotador %d: Empacotando pedido %d...\n", id, id_ep);
    empacotamento_concluido++;
    pthread_cond_signal(&cond_entregar);
    pthread_mutex_unlock(&mutex_ent);
    sleep(1);
```

O entregador_entregas, o estágio final do programa, é ativado após o empacotamento dos produtos. No entanto, ele verifica se há alguém buscando cartas antes de iniciar suas entregas. Quando pode prosseguir, ele adquire o lock de entrega para reduzir a contagem de empacotamentos concluídos. Caso o valor seja 0, ele entra em estado de espera novamente

```c
while (1) {
    // Esperar empacotamento e espera não estar buscando cartas .
    pthread_mutex_lock(&mutex_ent);
    while (empacotamento_concluido == 0 || buscando != 0) {
        pthread_cond_wait(&cond_entregar, &mutex_ent);
    }
    entregando++;
    id_e++;
    printf("Entregador %d: Entregando pedido %d...\n", id, id_e);
```

Neste programa, os processos de entrega podem ocorrer simultaneamente com os de produção e empacotamento. Para garantir essa sincronização, há dois locks separando e organizando os processos.

O lock de entrega tem como principal funcionalidade a manipulação e controle de variáveis relacionadas ao entregador, e garante que ambos os processos possam ocorrer simultaneamente, sem interferências ou inconsistências. Já o lock de produção é focado em manipular as variáveis de produção e empacotamento, ele assegura que essas operações sejam realizadas de forma ordenada, respeitando as dependências entre os processos.

```c
// Mutexes para controlar produção e entrega
pthread_mutex_t mutex_prod = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex_ent = PTHREAD_MUTEX_INITIALIZER;
```

Conclusão:

A implementação do programa demonstrou a importância do uso da programação concorrente para acelerar e manipular os dados de maneira mais eficiente, mesmo este sendo um problema fictício e simplificado, ele demonstrou vários dos desafios de uma possível empresa de entregas real, as quais dominam a realidade atual. Além disso, mostrou a importância do controle das variáveis com o uso de *locks* e variáveis de condição, o qual foi vital para a todos os processos do programa, impedindo a substituição de valores e *starvation*.

O controle de processos simultâneos na atualidade é vital para que possamos acelerar nossos processos e otimizá-los também, e este trabalho mostrou de maneira eficaz o funcionamento desses processos simultâneos em um caso real.

Referências:

ALCHIERI, Eduardo. *Locks*. Apresentação de slides. Brasília: Universidade de Brasília, 2025

ALCHIERI, Eduardo. Variáveis de condição. Apresentação de slides. Brasília: Universidade de Brasília, 2025