Univali – Escola Politécnica

João Victor Silvério

Sistemas Operacionais:

Trabalho M1 – IPC, Threads e Paralelismo

Ciência da Computação

Felipe Viel

Problema:

Diante disso, a empresa precisa que vocês implementem, por meio de aplicação para distribuição Linux/Windows, uma solução que consiga realizar as contagens nas três esteiras e exiba o resultado total (contagem esteira 1 + contagem esteira 2 + contagem esteira 3). A empresa pede que seja simulado a solução em um sistema multicore com sistema operacional com suporte a threads e IPC. A empresa solicita que um processo seja responsável pela contagem usando threads e outro processo seja responsável pela apresentação no display. Você deve usar pipe para que os dois processos troquem dados.

Além disso, os pesos dos itens que passam por cada uma das esteiras são armazenados em um único vetor de dados. A cada 1.500 unidades de produtos, das três esteiras, é necessário atualizar o peso total de itens processados. Sendo assim, a empresa aceita uma pausa na quantidade de itens sendo contados e pesados para realizar a pesagem total.

A empresa também fornece uma análise das três esteiras:

• Esteira 1: produtos de maior peso (5 Kg) – passa 1 item a cada segundo pelo sensor.

• Esteira 2: produtos de peso médio (2 Kg) – passa 1 item a cada 0,5 segundo pelo sensor.

• Esteira 3: produtos de menor peso (0,5 Kg) – passa 1 item a cada 0,1 segundo pelo sensor.

• A contagem deve acontecer initerruptamente.

• A exibição no display deve atualizar a cada 2 segundos para os operadores poderem acompanhar.

• Um operador pode usar um botão no equipamento para poder parar a contagem devido a um problema ocorrido.

Solução:

Criar uma aplicação utilizando Pthreads e IPC entre processos com pipe, de tal forma que, uma aplicação seja responsável pela utilização de threads, com uso de mutex para controle de acesso as seções críticas, e a outra, responsável pelo display das informações e saídas desejadas.

Portanto, foram criados dois arquivos em linguagem c, para que fosse possível resolver o problema via terminal. O primeiro sendo o servidor do pipe, funcionando como o display do problema, no qual mostraria os resultados obtidos através das esteiras.

E o segundo sendo o cliente do pipe, utilizando threads com o auxílio da biblioteca Pthreads, para dividir o controle das esteiras em 3 (uma thread para

cada), além de contagem do peso total depois de contabilizados 1500 itens, e a leitura do teclado para uma possível para manual da contagem.

Códigos:

Servidor:

```
/ Create socket
        sockfd = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sockfd < 0)
        {
            perror("Falha em criar o pipe");
            break;
        }

        // Bind socket to local address
        memset(&local, 0, sizeof(local));
        local.sun_family = AF_UNIX;
        strncpy(local.sun_path, SOCK_PATH, sizeof(local.sun_path) - 1);
        unlink(local.sun_path);
        len = strlen(local.sun_path) + sizeof(local.sun_family);
        if (bind(sockfd, (struct sockaddr *)&local, len) < 0)
        {
            perror("Falha em capturar o socket");
            close(sockfd);
            break;
        }

        // Listen for connections
        if (listen(sockfd, 5) < 0)
        {
            perror("Falha em escutar o socket");
            close(sockfd);
            break;
        }

        // Accept connections
        memset(&remote, 0, sizeof(remote));
        len = sizeof(remote);
        newsockfd = accept(sockfd, (struct sockaddr *)&remote, &len);
        if (newsockfd < 0)
        {
            perror("Falha em aceitar coneccao");
            close(sockfd);
            break;
        }

        printf("Cliente conectado!\n");

        // Read data from client
        if (read(newsockfd, buffer, sizeof(buffer)) < 0)
        {
            perror("Falha em ler do socket");
            close(newsockfd);
            close(sockfd);
            break;
        }

        printf("%s\n", buffer); //Imprime string passada pelo cliente
```

Cliente: Criação da thread de esteira (exemplo da esteira 1)

```c
thread_mutex_t exclusao_mutua = PTHREAD_MUTEX_INITIALIZER;

void *thread_sensor_1(void *param){    //Esteira 1
   while(1){
        if(stop!= 9){   //Se o botao de parada nao estiver sido adicionado
                pthread_mutex_lock(&exclusao_mutua);
                peso[contagem] = peso_esteira_1;
                itens++;
                contagem++;
                pthread_mutex_unlock(&exclusao_mutua);

                usleep(1000000); //sleep for 1 s
        }
        else{}   //Se o programa estiver parado
    }
}
```

Cliente: Criação das threads de contagem de peso e leitura de teclado

```c
void *contar_peso(void *param){
    while(1){
        if(contagem>=1500){
            pthread_mutex_lock(&exclusao_mutua);
            for(int i=0; i<contagem;i++){
                total += peso[i];
            }
            contagem = 0;
            pthread_mutex_unlock(&exclusao_mutua);
        }
    }
}

void *ler_teclado(void *param){
    while(1){
        scanf("%d", &stop); //Verifica se o botao de parada foi acionado
    }
}
```

Cliente: Criação da thread de display utilizando pipe

```c
void *display(void *param){  //Funcao de display usando pipe
    int sockfd, len;
    struct sockaddr_un remote;
    char item[1024];

    while(1){
        // Create socket
        sockfd = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sockfd < 0)
        {
            perror("Falha em criar o socket");
            break;
        }

        // Connect to server
        memset(&remote, 0, sizeof(remote));
        remote.sun_family = AF_UNIX;
        strncpy(remote.sun_path, SOCK_PATH, sizeof(remote.sun_path) - 1);
        len = strlen(remote.sun_path) + sizeof(remote.sun_family);
        if (connect(sockfd, (struct sockaddr *)&remote, len) < 0)
        {
            perror("Falha em conectar no servidor");
            close(sockfd);
            break;
        }

        // Send data to server
        pthread_mutex_lock(&exclusao_mutua);
        sprintf(item, "Total: %d e Peso: %f", itens, total); //contagem de itens e peso total
        pthread_mutex_unlock(&exclusao_mutua);
        if (write(sockfd, item, strlen(item) + 1) < 0)
        {
            perror("Falha em escrever no socket");
            close(sockfd);
            break;
        }

        // Read data from server
        if (read(sockfd, item, sizeof(item)) < 0)
        {
            perror("Falha em ler do socket");
            close(sockfd);
            break;
        }

        // Close socket and exit
        close(sockfd);
        usleep(2000000); //sleep for 2 s
    }
}
```

Cliente: Criação e junção das threads

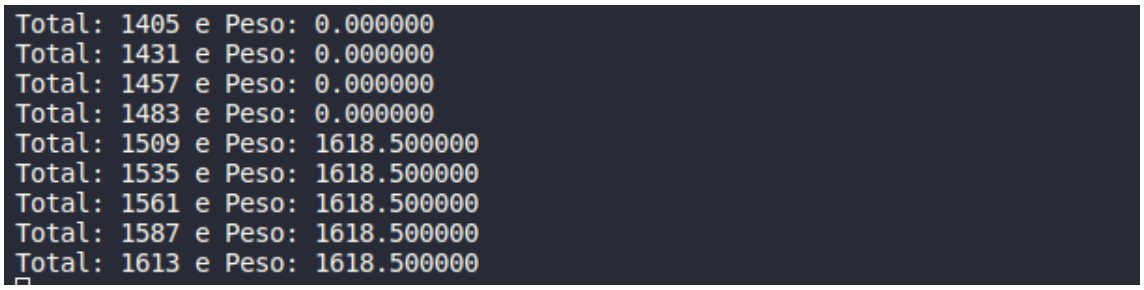
```c
void main(){
    //declarar variáveis e tipos para threads
    pthread_t tid_1, tid_2, tid_3, tid_display, tid_total_peso, tid_teclado;
    pthread_attr_t attr;

    pthread_attr_init(&attr);

    //criar thread de display com pipe
    pthread_create(&tid_display,&attr,display,NULL);

    //criar thread 1
    pthread_create(&tid_1,&attr,thread_sensor_1,NULL);

    //criar thread 2
    pthread_create(&tid_2,&attr,thread_sensor_2,NULL);

    //criar thread 3
    pthread_create(&tid_3,&attr,thread_sensor_3,NULL);

    //criar thread de contagem de peso
    pthread_create(&tid_total_peso,&attr,contar_peso,NULL);

    //criar thread leitura de teclado
    pthread_create(&tid_teclado,&attr,ler_teclado,NULL);

    //join thread
    pthread_join(tid_display, NULL);
    pthread_join(tid_1, NULL);
    pthread_join(tid_2, NULL);
    pthread_join(tid_3, NULL);
    pthread_join(tid_total_peso, NULL);
    pthread_join(tid_teclado, NULL);
}
```

Resultados:

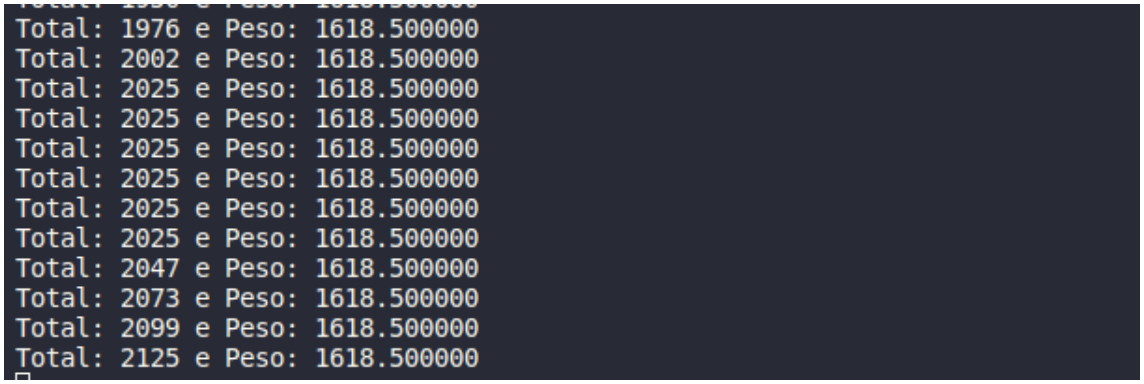
Primeiras saídas mostradas a cada 2s de contagem de todas as esteiras

```
silverio@silverio:~/Desktop/Materias/Sistemas-Operacionais/M1$ ./server
Total: 1 e Peso: 0.000000
Total: 27 e Peso: 0.000000
Total: 53 e Peso: 0.000000
Total: 79 e Peso: 0.000000
Total: 105 e Peso: 0.000000
Total: 131 e Peso: 0.000000
Total: 157 e Peso: 0.000000
Total: 183 e Peso: 0.000000
```

Saídas mostradas quando chegamos a contagem de 1500



Saídas mostradas enquanto o programa estiver pausado e depois do resumo do programa



Análise e discussão:

Utilizando da estratégia das threads, do uso de mutex para o controle de acesso e do uso do condicional while(1), para que o programa pudesse ficar em loop enquanto estiver rodando, podemos perceber que a contagem é realizada em duas formas. Uma para a contagem total dos itens passados na esteira, e outra para a contagem até o valor de 1500, para que possa ser feita a soma total do peso das unidades que já foram computadas em um vetor com esse tamanho.

Com essas variáveis calculadas, usa-se da thread de display para enviar ao outro lado do pipe(servidor) uma cadeia de caracteres com o total contabilizado e o peso somado a cada 1500 unidades. De tal forma que o servidor ao escutar o envio desta cadeia de caracteres a imprima na tela para o usuário.

É possível notar também o condicional dentro das threads de cada esteira, pois o programa precisa que estas parem de funcionar quando for apertado um determinado botão, fazendo com que se este estiver apertado a thread continuará funcionando porém sem realizar nenhuma ação, para não comprometer a contagem dos itens.

Códigos disponíveis em: https://github.com/Joao-Silverio/Sistemas-Operacionais.git