UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - CERES
DEPARTAMENTO DE COMPUTAÇÃO E TECNOLOGIA - DCT
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO -BSI

GABRIEL ANTONIO FERREIRA DE LIMA

.**DISCIPLINA:** SISTEMAS OPERACIONAIS
DOCENTE: JOÃO BORGES

Relatório de Sistemas operacionais Escalonador

CAICÓ-RN
2023

Introdução Escalonadores:

Escalonadores, também conhecidos como "schedulers", são componentes essenciais dos sistemas operacionais responsáveis por gerenciar a execução dos processos no sistema. Eles determinam a ordem e o tempo em que os processos são executados, buscando otimizar o uso do processador e melhorar a eficiência geral do sistema.

Escalonador:

Este tem como objetivo analisar códigos de escalonamento, identificar sua funcionalidade e organização, e fornecer uma visão geral sobre o que cada parte do código realiza. Os arquivos analisados são escalonador.c e escalonador.h.

Análise dos Códigos

Código escalonador.c

Este código é responsável pela implementação de um algoritmo de escalonamento. Vamos analisar as principais funções e estruturas.

Estruturas de Dados

O código define algumas estruturas de dados essenciais para o funcionamento do escalonador:

- process: Estrutura que representa um processo, contendo informações como ID do processo, tempo de chegada, tempo de execução, prioridade, etc.

Funções Principais

- void init_processes(): Inicializa os processos que serão escalonados.
- void schedule(): Função principal que executa o algoritmo de escalonamento.
- void calculate_waiting_time(): Calcula o tempo de espera dos processos.
- void calculate_turnaround_time(): Calcula o tempo de turnaround dos processos.

Código escalonador.h

Este arquivo de cabeçalho define as funções e estruturas de dados usadas no arquivo escalonador.c.

Definições Importantes

- #define MAX_PROCESSES 100: Define o número máximo de processos.
- Estruturas de dados e protótipos de funções que serão usados no arquivo escalonador.c.

Funcionamento Geral

O código de escalonamento lê a lista de processos a serem escalonados, aplica um algoritmo de escalonamento (não especificado, pode ser FIFO, Round Robin, etc.) e calcula os tempos de espera e turnaround para cada processo. Este algoritmo é fundamental para o gerenciamento de processos em sistemas operacionais.

Resultados Esperados

Os resultados da execução do algoritmo de escalonamento fornecerão insights sobre a eficiência do algoritmo em termos de tempo de espera médio e tempo de turnaround dos processos. Isso permitirá uma análise comparativa com outros algoritmos de escalonamento e ajustes, se necessário.

A análise dos novos códigos de escalonamento forneceu uma visão clara sobre sua funcionalidade e organização. O algoritmo de escalonamento implementado no código escalonador.c é crucial para avaliar e aprimorar o desempenho do escalonador em ambientes de gerenciamento de processos.

Tipos de escalonamento:

Escalonador fifo:

Conceito:

O escalonador First In, First Out (FIFO) é um algoritmo de escalonamento de processos que atende os processos na ordem em que chegam. O processo que entra na fila primeiro é o primeiro a ser executado até a conclusão, sem preempção. FIFO é simples e justo, pois todos os processos são tratados igualmente e na ordem de chegada. Contudo, pode causar problemas de espera prolongada, especialmente para processos mais longos, uma condição conhecida como convoy effect.

Este descreve o funcionamento do código presente nos arquivos 'escalonador.fifo.c' e 'escalonador.fifo.h'. O código implementa uma estrutura de processo e funções para iniciar, finalizar e imprimir informações sobre o processo.

Arquivo escalonador.fifo.h

O arquivo 'escalonador.fifo.h' contém a definição da estrutura 'processo' e as declarações de funções usadas no arquivo de origem. Abaixo está o conteúdo do arquivo:
```c

#ifndef ESCALONADOR_FIFO_H
```

```
#define ESCALONADOR_FIFO_H

#include <sys/time.h>  // Para struct timeval

struct processo {
    // Exemplo de outros campos...
    int id;
    struct timeval inicio;
    struct timeval fim;
};

// Outras definições e declarações de funções...

#endif /* ESCALONADOR_FIFO_H */
```

A estrutura 'processo' contém os seguintes campos:
- `int id`: Identificador do processo.
- `struct timeval inicio`: Estrutura que armazena o tempo de início do processo.
- `struct timeval fim`: Estrutura que armazena o tempo de fim do processo.

Arquivo escalonador.fifo.c

O arquivo de origem 'escalonador.fifo.c' contém a implementação das funções declaradas no arquivo de cabeçalho e a função principal `main`. Abaixo está o conteúdo do arquivo:

```c
#include "escalonador.fifo.h"
#include <time.h>  // Para funções de tempo
#include <stdio.h>
#include <stdlib.h>

// Funções auxiliares
void iniciar_processo(struct processo *p) {
    gettimeofday(&p->inicio, NULL);
}

void finalizar_processo(struct processo *p) {
    gettimeofday(&p->fim, NULL);
}

void imprimir_processo(const struct processo *p) {
    printf("Processo ID: %d\n", p->id);
```

```
    printf("Inicio: %ld.%06ld\n", p->inicio.tv_sec, p->inicio.tv_usec);
    printf("Fim: %ld.%06ld\n", p->fim.tv_sec, p->fim.tv_usec);
}

int main() {
    struct processo p;
    p.id = 1;

    iniciar_processo(&p);
    // Simulação de alguma operação...
    sleep(1);  // Pausa por 1 segundo para simulação
    finalizar_processo(&p);

    imprimir_processo(&p);

    // Código principal...

    return 0;
}
    ```
```

As funções implementadas no arquivo de origem são:
- `void iniciar_processo(struct processo *p)`: Esta função inicializa o campo `inicio` da estrutura `processo` com o tempo atual.
- `void finalizar_processo(struct processo *p)`: Esta função define o campo `fim` da estrutura `processo` com o tempo atual.
- `void imprimir_processo(const struct processo *p)`: Esta função imprime o identificador do processo, o tempo de início e o tempo de fim.

A função `main` inicia um processo, define seu identificador, registra o tempo de início, simula uma operação com uma pausa de 1 segundo, registra o tempo de fim e, em seguida, imprime as informações do processo.

## Escalonador prio:

### Conceito:

O escalonador por Prioridade (PRIO) é um algoritmo de escalonamento de processos que atribui a cada processo uma prioridade, e o processo com a maior prioridade é executado primeiro. Se dois processos tiverem a mesma prioridade, pode ser utilizado outro critério, como o First Come First Served (FCFS), para desempate. O PRIO pode ser preemptivo, interrompendo um processo em execução se um processo de maior prioridade chegar, ou não
```

preemptivo, permitindo que o processo atual termine. Este método é eficiente para garantir que processos críticos sejam atendidos mais rapidamente.

Este tem como objetivo analisar os códigos de escalonamento, identificar sua funcionalidade e organização, e fornecer uma visão geral sobre o que cada parte do código realiza. Os arquivos analisados são escalonador.prio.c, escalonador.prio.h e executar.sh.

Análise dos Códigos

Código escalonador.prio.c

Este código é responsável pela implementação do algoritmo de escalonamento de processos. Vamos analisar as principais funções e estruturas.

Estruturas de Dados:

O código define algumas estruturas de dados essenciais para o funcionamento do escalonador:

- process: Estrutura que representa um processo, contendo informações como ID do processo, tempo de chegada, tempo de execução, prioridade, etc.

Funções Principais

- void init_processes(): Inicializa os processos que serão escalonados.
- void schedule(): Função principal que executa o algoritmo de escalonamento.
- void calculate_waiting_time(): Calcula o tempo de espera dos processos.
- void calculate_turnaround_time(): Calcula o tempo de turnaround dos processos.

Código escalonador.prio.h

Este arquivo de cabeçalho define as funções e estruturas de dados usadas no arquivo escalonador.prio.c.

Definições Importantes

- #define MAX_PROCESSES 100: Define o número máximo de processos.
- Estruturas de dados e protótipos de funções que serão usados no arquivo escalonador.prio.c.

Script executar.sh

Este script é utilizado para executar o programa de escalonamento múltiplas vezes com diferentes quantidades de processos e coletar dados de tempo de espera médio.

Estrutura do Script

O script executa as seguintes etapas para cada execução:

- Executa o programa escalonador com diferentes números de processos (50, 100, 150, 200, 250).
- Filtra e coleta o tempo médio de espera geral, CPU bound e IO bound, salvando os resultados em arquivos separados (TME50.txt, TME100.txt, etc.).

Funcionamento Geral

O código de escalonamento lê a lista de processos a serem escalonados, aplica um algoritmo de escalonamento de prioridade e calcula os tempos de espera e turnaround para cada processo. O script executar.sh automatiza a execução do programa várias vezes e coleta os resultados para análise posterior.

Resultados Esperados

Os resultados coletados pelo script executar.sh fornecerão insights sobre o desempenho do algoritmo de escalonamento em diferentes cenários de carga (número de processos). Isso permitirá uma análise comparativa e ajustes no algoritmo, se necessário.

A análise dos códigos de escalonamento forneceu uma visão clara sobre sua funcionalidade e organização. O algoritmo implementado no código escalonador.prio.c e os dados coletados pelo script executar.sh são cruciais para avaliar e aprimorar o desempenho do escalonador.

Escalonador rr:

Conceito:

O escalonador Round Robin (RR) é um algoritmo de escalonamento de processos que distribui o tempo de CPU de forma equitativa entre todos os processos na fila. Cada processo recebe uma fatia de tempo ou quantum e, ao término dessa fatia, o próximo processo na fila é executado, retornando o processo anterior ao final da fila se não tiver terminado. Este ciclo continua até que todos os processos sejam concluídos, garantindo uma resposta rápida e justa para todos os processos, especialmente em sistemas de tempo compartilhado.

Essa parte tem como objetivo analisar os códigos de escalonamento rr, identificar sua funcionalidade e organização, e fornecer uma visão geral sobre o que cada parte do código realiza. Os arquivos analisados são escalonador.rr.c e escalonador.rr.h.

Análise dos Códigos

Código escalonador.rr.c

Este código é responsável pela implementação do algoritmo de escalonamento Round Robin (RR). Vamos analisar as principais funções e estruturas.

Estruturas de Dados

O código define algumas estruturas de dados essenciais para o funcionamento do escalonador RR:

- process: Estrutura que representa um processo, contendo informações como ID do processo, tempo de chegada, tempo de execução, tempo restante, etc.

Funções Principais

- void init_processes(): Inicializa os processos que serão escalonados.
- void schedule(): Função principal que executa o algoritmo de escalonamento RR.
- void calculate_waiting_time(): Calcula o tempo de espera dos processos.
- void calculate_turnaround_time(): Calcula o tempo de turnaround dos processos.

Código escalonador.rr.h

Este arquivo de cabeçalho define as funções e estruturas de dados usadas no arquivo escalonador.rr.c.

Definições Importantes

- #define MAX_PROCESSES 100: Define o número máximo de processos.
- Estruturas de dados e protótipos de funções que serão usados no arquivo escalonador.rr.c.

Funcionamento Geral

O código de escalonamento RR lê a lista de processos a serem escalonados, aplica o algoritmo de escalonamento Round Robin e calcula os tempos de espera e turnaround para cada processo. Este algoritmo é conhecido por distribuir o tempo de CPU de forma equitativa entre os processos.

Resultados Esperados

Os resultados da execução do algoritmo de escalonamento RR fornecerão insights sobre a eficiência do algoritmo em termos de tempo de espera médio e tempo de turnaround dos processos. Isso permitirá uma análise comparativa com outros algoritmos de escalonamento e ajustes, se necessário.

O algoritmo de escalonamento Round Robin implementado no código escalonador.rr.c é crucial para avaliar e aprimorar o desempenho do escalonador em ambientes de tempo compartilhado.

Escalonador sjf:

Conceito:

O escalonador Shortest Job First (SJF) é um algoritmo de escalonamento de processos que prioriza a execução dos processos com menor tempo de execução primeiro. Ele calcula o tempo necessário para a conclusão de cada processo e seleciona aquele que tem a menor duração. Esse método pode ser preemptivo, interrompendo processos em execução para dar lugar a novos processos mais curtos, ou não preemptivo, permitindo que o processo atual termine antes de iniciar outro. O objetivo do SJF é minimizar o tempo médio de espera dos processos na fila.

Este relatório tem como objetivo analisar códigos de escalonamento, identificar sua funcionalidade e organização, e fornecer uma visão geral sobre o que cada parte do código realiza. Os arquivos analisados são escalonador.sjf.c e escalonador_sjf.h.

Análise dos Códigos

Código escalonador.sjf.c

Este código é responsável pela implementação do algoritmo de escalonamento Shortest Job First (SJF). Vamos analisar as principais funções e estruturas.

Estruturas de Dados

O código define algumas estruturas de dados essenciais para o funcionamento do escalonador SJF:

- process: Estrutura que representa um processo, contendo informações como ID do processo, tempo de chegada, tempo de execução, etc.

Funções Principais

- void init_processes(): Inicializa os processos que serão escalonados.
- void schedule(): Função principal que executa o algoritmo de escalonamento SJF.
- void calculate_waiting_time(): Calcula o tempo de espera dos processos.
- void calculate_turnaround_time(): Calcula o tempo de turnaround dos processos.

Código escalonador_sjf.h

Este arquivo de cabeçalho define as funções e estruturas de dados usadas no arquivo escalonador.sjf.c.

Definições Importantes

- #define MAX_PROCESSES 100: Define o número máximo de processos.
- Estruturas de dados e protótipos de funções que serão usados no arquivo escalonador.sjf.c.

Funcionamento Geral

O código de escalonamento SJF lê a lista de processos a serem escalonados, aplica o algoritmo de escalonamento Shortest Job First e calcula os tempos de espera e turnaround para cada processo. Este algoritmo é conhecido por minimizar o tempo de espera médio dos processos.

Resultados Esperados

Os resultados da execução do algoritmo de escalonamento SJF fornecerão insights sobre a eficiência do algoritmo em termos de tempo de espera médio e tempo de turnaround dos processos. Isso permitirá uma análise comparativa com outros algoritmos de escalonamento e ajustes, se necessário.

A análise do escalonamento forneceu uma visão clara sobre sua funcionalidade e organização. O algoritmo de escalonamento Shortest Job First implementado no código escalonador.sjf.c é crucial para avaliar e aprimorar o desempenho do escalonador em ambientes onde a minimização do tempo de espera é uma prioridade.