**— Análise de Escalonadores**

**Título**

Comparação de Algoritmos de Escalonamento de Processos em Simulador Linux-Like

**Autores**

Jadson Hipólito de Almeida

Curso: Bacharelado em Sistemas de Informação, UFRN/CERES

Email: [[jadsonhipolito@gmail.com](mailto:seuemail@exemplo.com)]

**Resumo**

Este trabalho apresenta a implementação e análise comparativa de seis algoritmos de escalonamento de processos com base em simulação computacional. Foram avaliados os tempos médios de espera (TME) por algoritmo em diferentes cargas de trabalho. O estudo visa compreender o desempenho relativo de cada estratégia de escalonamento no gerenciamento de processos. Os resultados evidenciam comportamentos distintos, com variações de desempenho associadas às características específicas de cada algoritmo.

**Palavras-chave:** Escalonamento de Processos, TME, Sistemas Operacionais, Simulação, Algoritmos.

**1. Introdução**

O gerenciamento de processos é uma das funções centrais de um sistema operacional. Dentre os diversos componentes, o escalonador é responsável por definir qual processo será executado pelo processador em um dado instante. Neste trabalho, implementamos cinco algoritmos de escalonamento em um simulador e realizamos uma análise quantitativa de seus desempenhos com base no tempo médio de espera (TME).

**2. Metodologia**

2.1 Ambiente de Simulação

A implementação baseou-se no simulador disponível no repositório . Foram utilizados os seguintes arquivos e funções:

* main.c, stats.c, sched.c, entre outros módulos auxiliares
* A função accounting() foi adaptada para gerar as métricas TME, TMB e TMT.

2.2 Algoritmos Implementados

* **FIFO**: Primeiro processo a chegar é o primeiro a ser executado.
* **SJF**: Prioriza processos com menor tempo restante.
* **LJF**: Prioriza processos com maior tempo restante.
* **PRIO STATIC**: Utiliza duas filas fixas conforme o tempo restante.
* **PRIO DYNAMIC**: Reorganiza processos entre filas conforme eventos.
* **PRIO DYNAMIC QUANTUM**: Realimenta com base no uso do quantum disponível.

2.3 Coleta e Análise de Dados

Para cada algoritmo, foram realizadas 10 execuções para os seguintes tamanhos de carga: 10 a 100 processos (em múltiplos de 10). Os resultados foram registrados via script shell e analisados com Python (pandas e matplotlib).

**3. Resultados e Discussão**

3.1 Gráfico: Tempo Médio de Espera

Gráfico TME

Gráfico 1 – Tempo médio de espera (TME) por algoritmo, com barras de erro representando o desvio padrão.

3.2 Análise

* O algoritmo **SJF** apresentou o menor TME na maioria dos cenários, como esperado, por priorizar processos curtos.
* **LJF**, por priorizar processos longos, teve desempenho inferior.
* Os algoritmos com prioridade dinâmica (**PRIO DYNAMIC** e **PRIO DYNAMIC QUANTUM**) obtiveram desempenho intermediário e estável.
* A variação no TME cresce com o número de processos em todos os algoritmos, mas o impacto é maior nas políticas menos eficientes (como LJF).
* Os algoritmos com múltiplas filas tendem a suavizar picos de espera, beneficiando cargas maiores.

**4. Conclusão**

Os resultados confirmam que a escolha do algoritmo de escalonamento impacta diretamente a eficiência do sistema. O SJF apresenta o melhor desempenho em TME, mas é de difícil aplicação prática por exigir conhecimento prévio do tempo de execução. Algoritmos com múltiplas filas e realimentação, como o PRIO DYNAMIC QUANTUM, demonstraram bom equilíbrio entre desempenho e adaptabilidade.

**Referências**

* Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.).
* <https://github.com/labepi/sched_sim>
* Sociedade Brasileira de Computação. *Modelo SBC para Publicação de Artigos*. Disponível em: <https://www.sbc.org.br/wp-content/uploads/2024/07/modelosparapublicaodeartigos.zip>