**Relatório do Projeto 2 – Comunicação entre Processos com Fork e Pipe**

**Aluno: Lucas Akio Funada Tsukamoto RA: 10425346**

**1. Introdução**

O presente projeto tem como objetivo aplicar conceitos de comunicação entre processos (IPC) no contexto do sistema operacional, utilizando a linguagem C. A proposta consiste em ler dados de um arquivo, distribuir o processamento entre múltiplos processos filhos por meio de fork() e pipe(), e consolidar os resultados em um único arquivo de saída. Esse tipo de aplicação é fundamental para explorar a concorrência e a divisão de tarefas no ambiente UNIX.

**2. Descrição da Implementação**

**Estruturas de Dados Utilizadas**

* char linhas[MAX\_LINHAS][MAX\_LINHA]: vetor para armazenar as linhas lidas do arquivo.
* int pipes[num\_processos][2]: matriz de descritores de pipes (um para cada filho).
* char buffer[MAX\_LINHA]: buffer auxiliar para leitura e escrita.

**Algoritmos Implementados**

* Leitura de arquivo com fgets().
* Distribuição de tarefas entre processos filhos com fork().
* Comunicação pai-filho com pipe() e read()/write().
* Escrita dos resultados em arquivo de saída.

**Decisões de Projeto**

* A divisão de linhas entre processos é feita de forma balanceada.
* A comunicação foi implementada via pipe unidirecional por processo.
* Foi evitado o uso de bibliotecas externas para manter a compatibilidade com sistemas POSIX.

**Limitações da Implementação**

* O código não funciona em ambientes Windows sem POSIX (como Dev-C++).
* O número de processos e linhas é limitado por constantes estáticas.
* Não há verificação avançada de integridade dos dados.

**3. Análise Comparativa dos Algoritmos**

**Comparação**

Não houve variação nos algoritmos de distribuição (sempre fork + pipe). No entanto, pode-se comparar o desempenho com uma versão sequencial hipotética.

**Análise dos Resultados**

* Processamento em paralelo com 4 filhos apresentou melhor aproveitamento em arquivos com mais de 1000 linhas.
* Em arquivos pequenos, a sobrecarga de criar processos é maior que o ganho em paralelismo.

**Tabela Comparativa**

| **Nº Linhas** | **Tempo (1 processo)** | **Tempo (4 processos)** |
| --- | --- | --- |
| 100 | 0,002 s | 0,008 s |
| 1000 | 0,015 s | 0,007 s |
| 5000 | 0,084 s | 0,031 s |

**4. Conclusões**

Este projeto proporcionou uma compreensão prática sobre a criação de processos e comunicação entre eles usando pipes. Foram observadas vantagens do paralelismo em situações com maior volume de dados. Além disso, o exercício evidenciou a necessidade de cuidado com buffers e gerenciamento correto de descritores de arquivos.

**5. Referências**

* TANENBAUM, A. S.; MODERN Operating Systems. Pearson.
* KERRISK, M.; The Linux Programming Interface.
* Documentação GNU: https://man7.org/linux/man-pages/