E-fólio A | Folha de resolução para E-fólio

Sistemas Operativos | 21111

DOCENTES: Paulo Shirley, Paulo Quaresma, Rúdi Gualter Oliveira, José Félix Póvoa
ANO LECTIVO: 2022-2023

NOME: Caroline Araujo de Souza Santos
Nº DE ESTUDANTE: 2102995
TURMA: 3
CURSO: Engenharia Informática
DATA DE ENTREGA: 08/04/2023

RELATÓRIO

O programa precisa gerar e ordenar uma lista de n bytes aleatórios (que devem ser fornecidos na linha de comandos) e imprimir em decimal na saída padrão.

Bibliotecas utilizadas:

1. <stdio.h>: Para realizar operações de entrada e saída padrão (função printf e fread);
2. <stdlib.h>: Para converter tipos de dados, alocação dinâmica de memória e manipulação de arquivos (exit, atoi e fclose);
3. <unistd.h>: Para fazer chamadas de sistema e comunicação com o sistema operacional (fork e das funções da família exec);
4. <sys/wait.h>: Controlar processos filhos com a função wait.

Código:

Main recebeu dois argumentos (o nome do programa e o número de bytes).

- Checagem de número de argumentos. É importante que o programa tenha apenas dois argumentos (nome do programa + número de bytes aleatórios). Se tiver apenas um argumento, nenhum argumento ou 3 ou mais argumentos, o programa apresentará uma mensagem de erro e terminará.

- Checagem se o argumento n do número de bytes é maior que zero. Se n for menor ou igual a zero, o programa apresentará uma mensagem de erro e terminará. Se não, o programa executará normalmente.

Aqui, precisei converter o vetor argv[1], que é uma string, para um inteiro para que pudesse comparar com zero. Para isso, utilizei a função da stdlib atoi().

- Caso as duas checagens anteriores tenham tido sucesso, passamos para o programa em si.

Utilizei o printf para imprimir o argumento (n – número de bytes) que foi fornecido pelo usuário na linha de comando.

O processo A imprime o seu PID e o PPID na tela (utilizei o getpid() e o getppid()).

O programa A, pai, cria primeiramente um processo filho B (para isso utilizei a chamada fork ()) e aguarda o término desse processo filho B (para isso utilizei a função wait()).

O processo B imprime o seu PID e o PPID na tela (utilizei o getpid() e o getppid()).

O processo B redireciona sua saída padrão para o arquivo “tmp.bin” usando a função freopen().

Verificação se a abertura do arquivo foi bem sucedida (com o if).

O processo B executa o comando head para gerar um número aleatório de bytes e escrevê-los no arquivo “tmp.bin” (para isso utilizei a função de sistema execlp).

Verificação de funcionamento do execlp (caso não funcione, aparece uma mensagem de erro – utilizei o printf pra imprimir na tela) e o exit para o programa encerrar em caso de não funcionamento.

- O processo A cria um outro processo filho - Processo C (para isso utilizei a função fork ()).

O processo C espera a finalização do processo B (utilizei a função waitpid ()).

O processo C imprime o seu PID e o PPID na tela (utilizei o getpid() e o getppid()).

O processo C redireciona a saída padrão para o arquivo “tmp.txt” (utilizei também a função freopen ()).

Verificação se a abertura do arquivo foi bem sucedida.

O processo C executa o comando hexdump no arquivo “tmp.bin” (para isso utilizei a função de sistema execl), listando em decimal os bytes do ficheiro tmp.bin para o ficheiro tmp.txt.

Verificação de funcionamento do execl (caso não funcione, aparece uma mensagem de erro – utilizei o printf pra imprimir na tela) e o exit para o programa encerrar em caso de não funcionamento.

- O processo A cria um outro processo filho - Processo D (para isso utilizei a função fork ()).

Aguarda o B e C terminarem (utilizei a função waitpid()).

O processo D imprime o seu PID e o PPID na tela (utilizei o getpid() e o getppid()).

Abertura do arquivo tmp.txt para leitura (para isso utilizei novamente o freopen()).

O processo D executa o comando sort no arquivo “tmp.txt” para classificar os números aleatórios gerados em ordem crescente (utilizei a função de sistema execlv).

Verificação de funcionamento do execlv (caso não funcione, aparece uma mensagem de erro – utilizei o printf pra imprimir na tela) e o exit para o programa encerrar em caso de não funcionamento.

- Imprimindo na tela.

Abri o arquivo utilizando a função fopen(), no modo de somente leitura.

Criei uma variável do tipo int chamada numeros.

Fiz uma varredura no arquivo utilizando a função EOF (end of file) e imprimi todos os números (para isso, utilizei o while).

Fechei os arquivos abertos com fclose().

Fim do programa e retorno (0).