Faculdade de Ciência da Computação - FACOM
Programa de Mestrado em Ciência da Computação
MC104 - Sistemas Operacionais

Lista de Exercícios #2 — Sistema de Arquivos

Ex. 01: Utilizando somente funções para leitura e escrita, considere o programa abaixo que copia um arquivo. Este programa pode ser caracterizado pelos seguintes aspectos:

- o programa lê de um dispositivo padrão e escreve em um dispositivo padrão, ou seja, assume-se que esses dispositivos tenham sido definidos na *shell*;
- muitas aplicações assumem que a entrada padrão é o descritor 0 e a saída padrão é o descritor 1; neste exemplo nós assumimos 02 nomes STDIN_FILENO e STDOUT_FILENO definidos no arquivo de cabeçalho <unistd.h>;
- o programa não executa a operação *close* nos arquivos de entrada e de saída, ao invés disso, ele utiliza do fato de que quando um processo é terminado o UNIX fecha todos os descritores de arquivo;
- o exemplo funciona tanto para arquivos texto quanto para arquivos binário, desde que não haja diferença entre os 02 núcleos do UNIX.

```
#include ``ourhdr.h''

#define BUFFSIZE 8192

int main( int argc, char* argv[] )
{
  int n;
  char buff[BUFFSIZE];

  while( (n = read(STDIN_FILENO, buff, BUFFSIZE)) > 0 )  {
    if( write(STDOUT_FILENO, buff, n) != 0 )
      err_sys(``write error'');
  }

  if( n < 0 )
    err_sys(``read error'');

  exit( 0 );
}
```

Como escolher o valor de BUFFSIZE ? Para responder a esta questão, execute o programa para diferentes valores de BUFFSIZE. Iniciando em 1 e utilizando a potência de 2, faça os valores variarem segundo a seqüência: 1, 2, 4, 8, 16, 32, 64, 128, até 16384. Para este teste, informe qual foi o tamanho do bloco do sistema de arquivo utilizado. Monte uma tabela

informando para cada BUFFSIZE, o tempo utilizado pela CPU e o número de *loops*. Analise os resultados e responda a pergunta inicialmente feita.

Ex. 02: Implemente o algoritmo *getblk* que tem como entrada dois números inteiros (sistema de arquivos e bloco do disco) e como saída um buffer já posicionado na tabela de *hash*. Considere um *pool* de 10 buffers de 512 bytes cada. Caso nenhum buffer esteja disponível, o algoritmo bloqueia indefinidamente. Numa situação real o bloqueio termina quando outro processo libera um ou mais buffers.

A simulação do dispositivo físico pode ser implementada com um arquivo de 10x512 bytes (para o caso de escrita assíncrona) e chamadas **write** e **lseek**.

Ex. 03: Simule o algoritmo para alocação e liberação de buffers até que ocorra a exaustão do *pool* de *buffers*.