Atividade Experimental 1

Implementação de biblioteca de filas – "*fila2*"

1 Descrição Geral

O objetivo desta atividade é implementar uma biblioteca para o tratamento de filas, duplamente encadeadas. As filas são estruturas de dados importantes em um sistema operacional e servem, entre outros usos, para organizar as estruturas de dados que representam os processos, existentes em um escalonador de curto prazo. Assim, por exemplo, sempre que um processo realiza uma transição de estados, o escalonador passa as estruturas de dados correspondentes ao processo de uma fila para outra.

A biblioteca "fila2" deve ser implementada segundo a interface listada nesse documento e conforme a descrição do arquivo *header* fornecido.

Essa atividade é INDIVIDUAL!

Essa atividade é OPCIONAL! Mas, assim como as demais "Atividades Experimentais", deverão ser realizadas para que o aluno possa utilizar as notas das atividades experimentais no cálculo da média final dos trabalhos.

As implementações entregues serão corrigidas por um CORRETOR AUTOMÁTICO!

2 Interface de Programação da biblioteca

As funções a serem implementadas estão resumidas na tabela 1. Na tabela são apresentados os tipos de dados e os protótipos de função que estão definidos no arquivo *fila2.h* fornecido junto com a especificação desta atividade. Esse arquivo NÃO DEVE SER ALTERADO, pois representa a interface entre a biblioteca e as aplicações que vão usá-la. Além disso, a implementação das funções, assim como os tipos de dados definidos nesse arquivo, devem seguir, RIGOROSAMENTE, o especificado.

A implementação da biblioteca deverá utilizar duas estruturas de dados: "*sFilaNode2*" e "*sFila2*", que estão definidas no arquivo *fila2.h*, junto com a declaração dos tipos "NODE2" e "PNODE2", para a primeira estrutura, e "FILA2" e "PFILA2", para a segunda estrutura, respectivamente.

A estrutura "*sFila2*" possui os elementos necessários para representar uma fila. Os elementos "*first*" e "*last*" apontam para uma estrutura do tipo "*sFilaNode2*", que representam o primeiro e o último nodo da fila, respectivamente. O elemento "*it*" é o "iterador" da fila, usado para "navegar" pelos elementos da mesma (ver funções *FirstFila2*, *LastFila2* e *NextFila2*).

A estrutura "*sFilaNode2*" possui os elementos necessários para representar um **item** (nodo) da fila. Nessa estrutura estão os ponteiros "*next*" e "*ant*", usados para ligar um nodo aos seus vizinhos "seguinte" e "anterior", respectivamente. Também está presente um ponteiro "*node*", usado para salvar o ponteiro dos dados armazenados nesse item da fila.

As estruturas de dados, conforme aparecem no arquivo "file2.h", são apresentadas na figura 1.

```
struct  sFilaNode2 {
        void    *node;                  // Ponteiro para a estrutura de dados do NODO
        struct  sFilaNode2 *ant;        // Ponteiro para o nodo anterior
        struct  sFilaNode2 *next;       // Ponteiro para o nodo posterior
};
struct sFila2 {
        struct  sFilaNode2 *it;         // Iterador para varrer a lista
        struct  sFilaNode2 *first;      // Primeiro elemento da lista
        struct  sFilaNode2 *last;       // Último elemento da lista
};

typedef struct sFilaNode2       NODE2;
typedef struct sFila2           FILA2;
typedef struct sFilaNode2 *     PNODE2;
typedef struct sFila2 *         PFILA2;
```

Figura 1 – Estruturas de dados que devem ser usadas na implementação

Nome		Descrição
int	CreateFila2 (PFILA2 *pFila*);	Inicializa uma estrutura de dados do tipo FILA2.
int	FirstFila2 (PFILA2 *pFila*);	Posiciona o iterador da fila para o primeiro item da mesma.
int	LastFila2 (PFILA2 *pFila*);	Posiciona o iterador da fila para o último item da mesma.
int	NextFila2 (PFILA2 *pFila*);	Posiciona o iterador da fila para o item seguinte àquele apontado pelo iterador antes da chamada da função.
void	*GetAtIteratorFila2 (FILA2 *pFila*);	Retorna o ponteiro armazenado no conteúdo do item endereçado pelo iterador da fila. É o elemento "nodo" da estrutura "sFilaNodo2".
int	AppendFila2 (PFILA2 *pFila*, void **content*);	Salva o ponteiro "*content*" em um novo item e coloca-o no final da fila "*pFila*". Requer alocação dinâmica da estrutura "sFilaNodo2".
int	InsertAfterIteratorFila2 (PFILA2 *pFila*, void **content*);	Coloca o ponteiro "*content*" em um novo item e coloca-o logo após o item apontado pelo iterador da fila "*pFila*". Requer alocação dinâmica dessa estrutura "sFilaNodo2".
int	DeleteAtIteratorFila2 (PFILA2 *pFila*);	Remove da fila "*pFila*" o item indicado pelo iterador e libera a memória correspondente à estrutura "sFilaNodo2". Ao final da função, o iterador deverá estar apontando para o elemento seguinte àquele que foi apagado. Vai receber NULL se a fila ficar vazia.

Tabela 1 – Interface de programação da biblioteca

2.1 Dicas de Operação

Algumas dicas e detalhes de operação da biblioteca:

1. Para criar uma fila é necessário declarar uma variável do tipo FILA2 e inicializar seu conteúdo, usando a função "*CreateFila2*".
2. Quando for solicitada a inserção de um novo item na fila (o que acontece nas funções "*AppendFila2*" e "*InsertAfterIteratorFila2*"), a biblioteca deve realizar a alocação dinâmica da estrutura "s*FilaNode2*", preenchê-la adequadamente, e colocá-la na fila.
3. Essa biblioteca é útil para armazenar filas de quaisquer tipos de dados. Isso é possível porque os dados a serem armazenados devem ser passados na forma de ponteiros do tipo (*void **). Ao inserir um item na fila deve-se realizar o *type-casting* para (*void **) e, ao ler um item da fila deve-se realizar o *type-casting* para a estrutura correta do item obtido.
4. A implementação da fila armazena apenas os ponteiros fornecidos. Ela não armazena os dados apontados. A área com esses dados deve ser mantida alocada pela aplicação que utiliza a biblioteca.
5. Observar o valor dos ponteiros *first* e *last*, que devem ser atualizados durante as operações com a pilha. Deve-se ter cuidado especial quando a fila estiver vazia e for acrescentado o primeiro item, assim como quando for removido um item que tornará vazia a fila.

3 Geração da *libfila2*

As funcionalidades para o gerenciamento de filas deverão ser disponibilizadas através de uma biblioteca de nome *libfila2.a*. Para gerar essa biblioteca deverá ser utilizada a seguinte "receita":

1. Cada arquivo "*<file_name.c>*" deverá ser compilado com a seguinte linha de comando:

```
gcc -c <file_name.c> -Wall
```

2. Para gerar a *fila2.a*, todos os arquivos compilados (representados por *file_1*, *file_2*, ...) devem ser ligados usando a seguinte linha de comando:

```
ar crs libfila2.a <file_1>.o <file_2>.o ...
```

Para ambas as etapas: compilação e geração da biblioteca, não deverão ocorrer mensagens de erro ou de "*warning*".

Para fins de teste, a biblioteca *libfila2.a* deverá ser ligada com o programa chamador (aplicação). Para fazer a ligação deve-se utilizar a seguinte linha de comando:

```
gcc -o <nome_exec> <nome_chamador.c> -L<lib_dir> -lfila2 -Wall
```

onde <*nome_exec*> é o nome do executável a ser gerado, <*nome_chamador.c*> é o nome do arquivo fonte onde é realizada a chamada das funções da biblioteca, <*lib_dir*> é o caminho do diretório onde está a bibliotecas "*libfila2.a*".

4 Entregáveis: o que deve ser entregue?

Devem ser entregues:

- Todos os arquivos fonte (arquivos ".c" e ".h") que formam a biblioteca "*libfila2*";
- Arquivo *makefile* para criar a "*libfils2.a*".
- O arquivo "*libfila2.a*" e

Os arquivos devem ser entregues em um *tar.gz* (NÃO USAR OUTROS FORMATOS), seguindo, **obrigatoriamente**, a seguinte estrutura de diretórios e arquivos:

\t2fs		
	include	DIRETÓRIO: local onde são postos todos os arquivo ".h". Nesse diretório deve estar o "*libfila2.h*"
	lib	DIRETÓRIO: local onde será gerada a biblioteca "*libfila2.a*".
	src	DIRETÓRIO: local onde serão postos todos os arquivos ".c" (códigos fonte) usados na implementação da "*libfila2.a*".
	makefile	ARQUIVO: arquivo makefile com regras para gerar a "*libfila2*". Deve possuir uma regra "*clean*", para limpar todos os arquivos gerados.

5 Avaliação

Para que um trabalho possa ser avaliado ele deverá cumprir com as seguintes condições:

- Entrega dentro dos prazos estabelecidos;
- Obediência à especificação: formato e nome das funções e estrutura de diretórios. Se não for obedecida essa determinação, o corretor automático indicará erro na implementação;
- Compilação e geração da biblioteca sem erros ou *warnings*;
- Fornecimento de todos os arquivos solicitados;

A biblioteca será corrigida de forma automática, e sua valoração será proporcional ao número de funções que estiverem operando corretamente.

6 Observações

Recomenda-se a troca de ideias entre os alunos. Entretanto, a identificação de cópias de trabalhos acarretará na aplicação do Código Disciplinar Discente e a tomada das medidas cabíveis para essa situação.