Support Library – Version 2018/1

1 Descrição Geral

A biblioteca *support* é composta por várias funções que auxiliam (dão suporte) na implementação dos trabalhos de "Sistemas Operacionais I-N" dos cursos de Ciência da Computação e Engenharia de Computação do Instituto de Informática da UFRGS.

Atualmente estão disponíveis três grupos de função:

- Uma função para obter a versão atual da biblioteca;
- Funções para medição de tempo;
- Funções para tratamento de filas;
- Uma função que fornece um número pseudo-aleatório;

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.

2 Função com a versão da biblioteca

Usada para identificar a versão da biblioteca.

Descrição da Interface

int **Version** (void)
Retorna número que identifica a versão da biblioteca. A versão é calculada da seguinte forma: (2 * ANO) + (SEMESTRE – 1)

3 Funções para medição de tempo

São duas funções, startTimer() e stopTimer(), que devem ser chamadas, respectivamente, antes do intervalo de tempo a ser medido e depois do intervalo de tempo a ser medido. Essas funções acessam os contadores de tempo do processador, que tem 64 bits. Entretanto, a função stopTimer() calcula a diferença entre um tempo e o outro e retorna um número de 32 bits sem sinal, correspondente ao tempo transcorrido entre a chamada da função startTimer() e a função stopTimer().

É importante salientar que os contadores de tempo do processador informam, na realidade, o número de ciclos de relógio desde que o processador foi ligado. Dessa forma, para se obter a informação real de tempo, é necessário dividir o valor informado por esses contadores pela frequência de relógio do processador. No Linux pode-se obter a informação da frequência do processador usando o comando "lscpu". Também é possível obter a mesma informação listando o arquivo "/proc/cpuinfo".

Abaixo é apresentado um exemplo de aplicação dessas duas funções:

```
unsigned int tempo;
startTimer();
// realiza a operação
tempo = stopTimer();
printf ("tempo transcorrido = %u\n", tempo);
```

Descrição da Interface

void **startTimer** (void)
Inicia a medição de tempo

unsigned int **stopTimer**(void)
Encerra a medição de tempo e retorna o número de ciclos do relógio da CPU transcorridos desde a última chamada de **startTimer**.

4 Funções de acesso e controle de filas duplamente encadeadas

Para gerenciar as filas, a biblioteca utiliza duas estruturas de dados: "*sFilaNode2*" e "*sFila2*", que estão definidas no arquivo "*support.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 "*support.h*", são apresentadas na figura 1.

O detalhe das funções da biblioteca podem ser encontrados no arquivo "*support.h*", inclusive o significado dos códigos de retorno das funções.

Alguns detalhes para utilização das funções de acesso às filas:

1. Para criar uma fila é necessário declarar uma variável do tipo FILA2 e inicializar seu conteúdo, usando a função "CreateFila2".
2. 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.
3. 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.

```
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

Descrição da Interface

int **CreateFila2** (PFILA2 pFila)
Inicializa uma estrutura de dados do tipo FILA2.

int **FirstFila2** (PFILA2 pFila)
Posiciona o iterador da fila no 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".

void ***GetAtNextIteratorFila2** (FILA2 pFila)
Retorna o ponteiro armazenado no conteúdo do PRÓXIMO (next) item endereçado pelo iterador da fila. É o elemento "nodo" da estrutura "sFilaNodo2->next".

void ***GetAtAntIteratorFila2** (FILA2 pFila)
Retorna o ponteiro armazenado no conteúdo do item ANTERIOR (ant) endereçado pelo iterador da fila. É o elemento "nodo" da estrutura "sFilaNodo2->ant".

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 da estrutura "sFilaNodo2".

int **InsertBeforeIteratorFila2** (PFILA2 pFila, void *content)
Coloca o ponteiro "*content*" em um novo item e coloca-o imediatamente antes do item apontado pelo iterador da fila "*pFila*". Requer alocação dinâmica da 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.

Na figura 2, a operação de algumas das funções anteriores estão representadas graficamente.

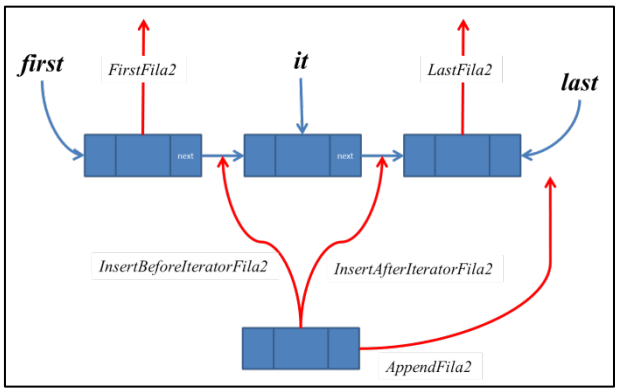

Figura 2 – Representação gráfica de algumas das funções da biblioteca

5 Função que fornece um número aleatório

Possui apenas uma função que fornece um número pseudo-aleatório. Para a geração desse número é utilizada uma semente 0xAA55 e o polinômio de geração $x^{16} + x^{14} + x^{13} + x + 1$.

Descrição da Interface

unsigned int **Random2**(void)
Gera um número pseudo-aleatório entre 0 e 65535

6 Sobre as Bibliotecas

Apesar desse documento empregar o nome "*support library*" e mencionar o termo *biblioteca*, as funções são oferecidas na forma de um arquivo binário. Formalmente, nos ambientes Unix e C, uma biblioteca é um tipo especial de arquivo com uma extensão *.a* (bibliotecas estáticas) e *.so* (bibliotecas dinâmicas). Os sistemas Windows também possuem bibliotecas estáticas e dinâmicas, essas últimas são mais conhecidas pelo acrônimo DLL (*Dynamic Link Library*). Neste documento, o termo biblioteca foi livremente empregado para fazer alusão a "um conjunto de funções".

Para utilizar as funções de *support.o* em seus programas é necessário ligar o arquivo *support.o* com os demais binários e bibliotecas que comporão o arquivo executável final. Essa ligação pode ser feita, por exemplo, com o seguinte comando:

gcc -o myprog.c support.o

Esse exemplo realize a compilação do programa myprog.c considerando que ele utilize uma ou mais funções providas pelo binário *support.o*. A execução do *gcc* com a opção *–o* gera o arquivo

executável *myprog* e realiza a ligação com *support.o*, adicionando a esse executável o corpo das funções de *support*.*o*.