Gabarito da AD2 de Programação I

Rio de Janeiro, 3 de junho de 2011

1. Funcionamento geral

O sistema proposto na AD2 deve seguir os mesmos padrões da aula 9 (relógio), mas deve incluir a execução de uma música, à escolha do usuário, durante o alarme. A tela principal do sistema deve ser algo semelhante ao esboço abaixo:

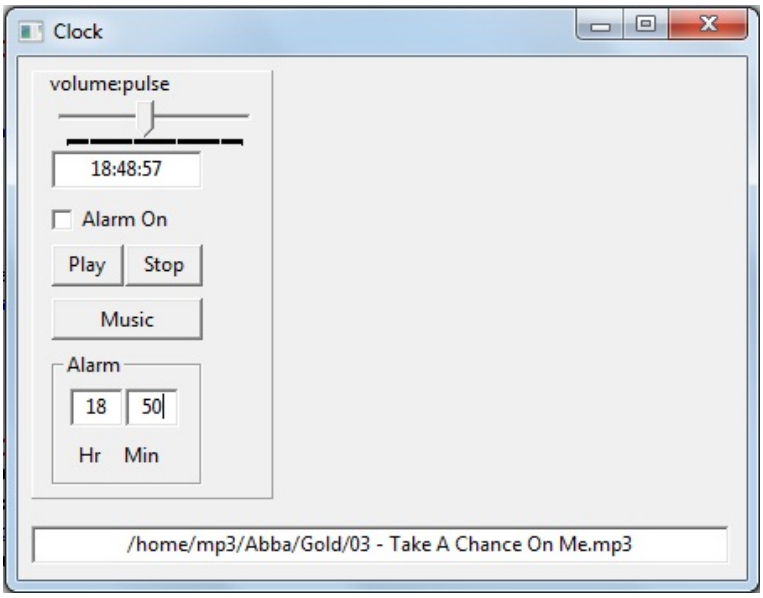

Figura 1: Tela principal

O formulário deve conter:

- Um **TTrackBar** para controlar o volume;
- Um **TEdit** para exibir a hora atual;
- Dois **TEdits** para inserir a hora e os minutos do alarme;
- Um **TSpeedButton** para abrir um **TOpenDialog** que servirá para escolher uma música;
- Dois **TButtons** para tocar e parar a música escolhida;

- Um **TEdit** para exibir o título da música escolhida;
- Uma forma de saber se o alarme está ligado ou não (pode ser um TCheckBox ou um TButton).

i) O volume atual (pode ser apenas o volume das caixas de som do computador, não precisa ser o volume do sistema operacional) deve ser manipulado pelo TTrackBar, de acordo com a posição da barra deslizante. Para isso, o sistema precisa ser capaz de executar um programa externo para ajustar o volume. Mais especificamente, é necessário executar um controlador de volume do sistema operacional. Isso é feito através de um **TProcess**.

A linha de comando que o TProcess responsável pelo gerenciamento do volume deve executar depende do sistema operacional e do controlador de volume escolhido. No caso do controlador de volume **amixer**, a linha de comando é a seguinte:

```
procedure TformRelogio.VolumeTrackBarChange(Sender: TObject);
  var
     v: Integer;
     s: string;
begin
  v := VolTrackBar.position;
  s := 'set PCM ' + IntToStr(v) + '%';
  VProcess.CommandLine := '/usr/bin/amixer -q -c 0 ' + s;
  (...)
end;
```

Explicando a linha de comando usada:

- -q: indica que mudanças durante a execução do amixer não devem ser visíveis;
- -c 0: indica qual placa de som será usada pelo amixer (0 indica a placa primária do computador);
- set PCM (...): é o tipo de volume que o amixer ajustará. PCM (*Pulse Code Modulation*), em Linux, engloba todos os sons que vêm de algum arquivo de som localizado em alguma unidade de memória do computador.

ii) A hora atual deve ser exibida e atualizada usando um **TTimer**, de forma semelhante ao que foi feito na aula 9 (relógio).

iii) O TSpeedButton será responsável pela escolha da música do alarme, chamando um **TOpenDialog** para tal. É importante que seja um TSpeedButton ao invés de

um TButton porque o TSpeedButton não permite mudança de foco ao ser pressionado, obrigando assim o usuário a escolher alguma música antes de fazer qualquer outra ação na interface do alarme. O título da música escolhida deve ser exibida em um TEdit. Abaixo, encontra-se uma visualização da utilização do TSpeedButton:

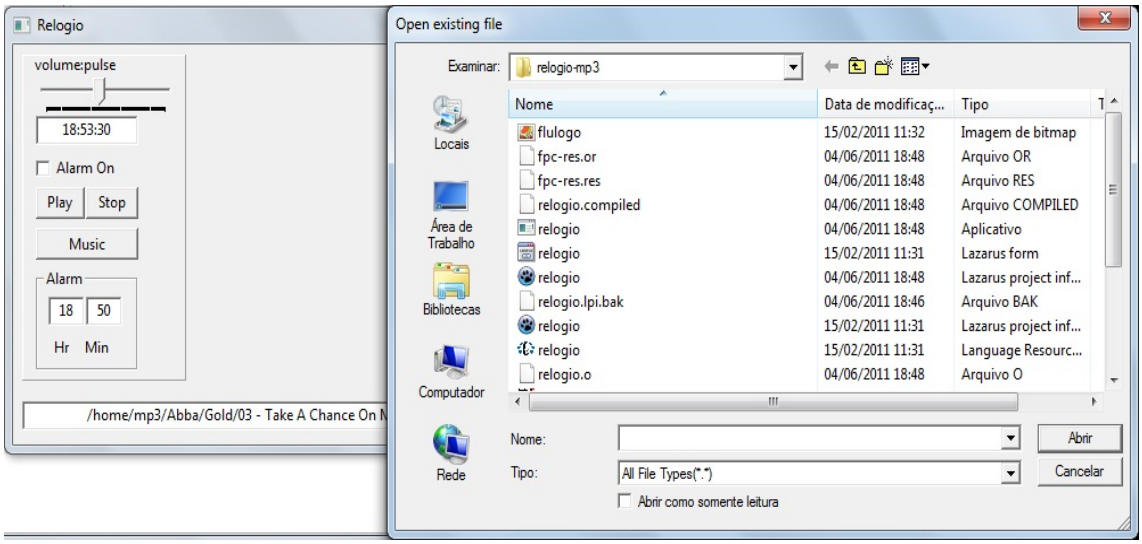

Figura 2: Utilização do TSpeedButton

iv) Os dois TButtons responsáveis por carregar e tocar a música escolhida no item anterior devem fazê-los através de outro TProcess, exclusivo para gerenciar um *player* de música a escolher. Logo abaixo encontra-se um exemplo de execução de música, usando o *mpg123*. Para parar a música, basta terminar o TProcess.

```
procedure TformRelogio.playMusic ( );

// txtSong  = TEdit contendo o título da música
// AProcess = TProcess criado para gerenciar o mpg123

var len: integer;
    ext: string;
    player, args: string;
begin
        (...)
        len := length(txtSong.text);
  ext := Copy(txtSong.text, len-2, 3);
  player := '/usr/bin/mpg123';
```

```
    if ( ext = 'm3u' ) then
        args := ' -C -@ '
    else
        args := ' -C ';
    AProcess.CommandLine := player + args + '"' +
            txtSong.text + '"';
    (...)
    end;
end;
```

Explicando a linha de comando usada:

- -C (opcional): permite que atalhos de teclado sejam usados para controlar a música (por exemplo, a música poderá ser parada pressionando "s");
- -@: indica que o próximo parâmetro é o endereço/URL da música a ser tocada na inicialização do *player*;

v) Por fim, o funcionamento do alarme estará condicionado a uma forma de controle a escolher (pode ser um TCheckBox ou um TButton).

Quando o alarme for ligado, o **TTimer** associado a ele deve ser "iniciado"através da atribuição do valor '0' ao seu atributo *Tag*, o nome do formulário deve passar a ser a hora nos TEdits de hora e minuto para o alarme disparar e a tela deve ser maximizada (semelhante a como foi feito na aula 9).

Caso o alarme seja desligado, o TTimer associado à animação deve ser interrompida, a música deve ser parada (terminando o TProcess do *player*) e a janela deve voltar a ter o tamanho normal.

Seguem abaixo duas visualizações do funcionamento do alarme: a primeira (Figura 3) é da ativação do alarme e a segunda (Figura 4) é do disparo.

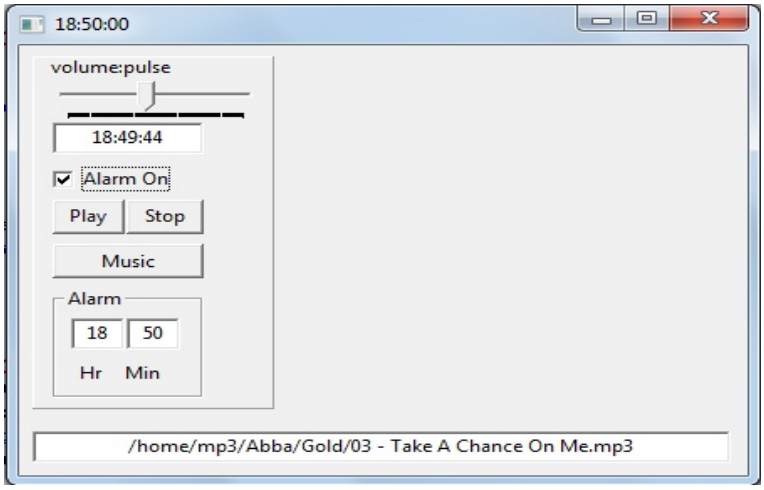

Figura 3: Ligando o alarme

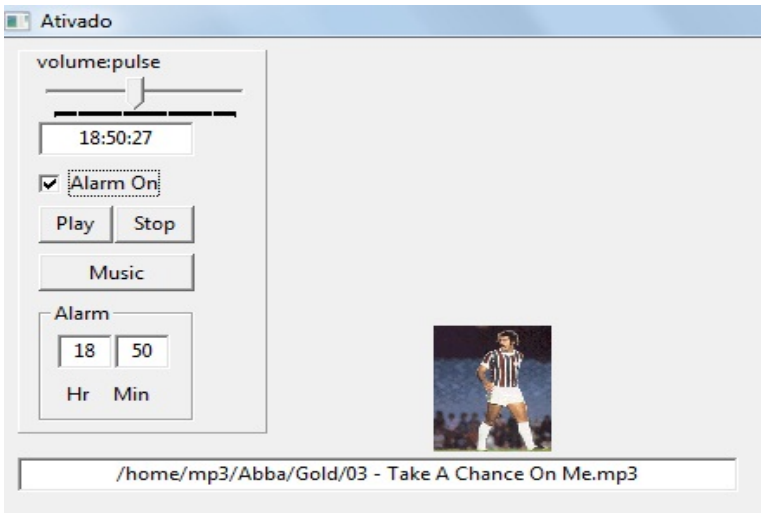

Figura 4: Disparo do alarme

2. Observações

Observação 1: Nenhuma funcionalidade do alarme deve interferir/interromper/abortar nenhuma outra funcionalidade. Para isso, deve-se tomar cuidado se tudo está funcionando como deveria e se todas as ações possíveis no sistema foram devidamente tratadas. Além disso, os processos precisam ser interrompidos quando o

sistema é encerrado. Para isso, não se pode esquecer de terminá-los ao encerrar a aplicação.

Observação 2: As imagens da animação são à escolha do aluno. O obrigatório é a execução correta da animação.

Observação 3: Validações nos TEdits de hora/minuto são opcionais, mas darão um acréscimo na nota de quem os fizer.