



Companhia Brasileira de Alumínio

# Treinamento Desenvolvimento de Aplicações no Sistema Score 7/8



High performance. Delivered.



accenture

consulting | technology | outsourcing

Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

## Objetivos e Pré-Requisitos

- Objetivos – Capacitar o Desenvolvedor a:
  - Entender a Arquitetura de Software do Score
  - Desenvolver e configurar uma aplicação completa
  - Instalar a Aplicação no ambiente de produção em segurança
- Pré-Requisitos
  - Domínio do sistema operacional e ambiente de desenvolvimento QNX
  - Conhecimento avançado de programação em linguagem C
- Plataforma de Software Requerida
  - Sistema Operacional QNX
  - Interface Gráfica Photon
  - Compilador Watcom C/C++ 10.6
  - TCP/IP Development Toolkit
  - Score Runtime
  - Score Development Toolkit

# Apresentação

---

## Tópicos Abordados no Treinamento

- Apresentação do Sistema
  - IHM – Interface Humano-Máquina
  - Principais Funcionalidades
- Ferramentas de Desenvolvimento:
  - Base de dados de Strings
    - # EditBdStrph
    - # LoadBdStr
  - Configuração de tarefas
    - # InstalaCtrl
  - Atualização arquivo DescrArqVar
    - # IniArqVar
  - Configuração da base de dados
    - # InstalaDemo
  - Configuração de eventos
    - # DescrEvph
    - # IniDescrEv
  - Configuração Descritor de Relatórios
    - # DescrRelph
  - Configuração da IHM
    - # ConfIHMph
  - Seleção da Base de Simulação
    - # selred

## Tópicos Abordados no Treinamento

- Estrutura de Dados:
  - Arquitetura
  - Tipos de Variáveis (parâmetros, variáveis de processo, diárias e turno)
  - Declaração de Eventos
- Configuração:
  - Parâmetros
  - Eventos
  - Descritor Relatórios
  - Chamada na IHM
- Programação em C:
  - Tarefas (Mcp e Mcc)
  - Programas de operação
  - Relatórios (ciclo de controle, diários e turno)

# Apresentação

---

## Tópicos Abordados no Treinamento

- Preparação e Instalação do Upgrade:
  - Backup dos arquivos que serão alterados
  - Geração do arquivo de upgrade
  - Instalação do upgrade
- Projeto Final
  - Desenvolvimento de um algoritmo:
    - Declaração de Variáveis
    - Tarefa Mcp
    - Tarefa Mcc
    - Programa de Operação
    - Relatório de Ciclo de Controle
    - Relatório Diário e Turno
  - Instalação do algoritmo nos nodos de produção

## Laboratório

- Partir o sistema de controle na máquina virtual QNX4\_VM-1 e a interface de supervisão na QNX4\_VM-3
- Apresentar os principais módulos e telas da IHM do Score
  - Pré-seleção
  - Configuração
  - Operação
  - Relatórios
    - Parâmetros de Linha e Cuba
    - Instantâneos
    - Ciclo de Controle e Históricos Diário/Turno
    - Eventos e Mensagens
    - Gráficos
      - Alarmes
      - Módulo Histórico

# Apresentação

---

## Laboratório (continuação ...)

- Apresentar alguns conceitos básicos da IHM
  - Base de dados de strings
  - Parâmetros de cuba e linha
  - Variáveis de processo
  - Eventos
  - Alarmes
  - Gráficos
- Preparar a máquina virtual para o desenvolvimento do sistema:
  - Instalar o runtime do Score (original CBA)
  - Instalar o compilador Watcom C e toolkit de desenvolvimento TCP/IP e photon
  - Instalar o arquivo **qmake.tar.F**
  - Instalar o toolkit de desenvolvimento **score\_kit\_des.tgz**
  - Instalar a base de dados de demonstração **cbabddemo.tgz**



Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings

**1º Passo** : carregar o photon

```
# ph
```

**2º Passo** : chamar o editor pela **Barra de Ferramentas** selecionando **Grupo Score Tools → EditBdStr**, ou via shell pelo comando abaixo:

```
# /score/util/EditBdStr
```



# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings

- A tela do programa é carregada em branco. Na régua do rodapé há botões de paginação, manipulação de informações e saída:



- **Botões Paginação:** **PGUP**, **PGDN**, **HOME**, **END**
- **Botões Manipulação:** tratamento de informações
  - **READ** : carregar a base de dados selecionada para memória do editor
  - **WRITE** : gravar os dados da memória para arquivo em disco
  - **GOTO** : posicionar na linha especificada
  - **SEARCH** : pesquisar o padrão de string especificado
  - **DIFF** : comparar as informações de memória com o arquivo selecionado
  - **COPY** : copiar informações do arquivo selecionado para a base de dados aberta em memória
  - **LOAD** : não utilizada
  - **QUIT** : sair do programa

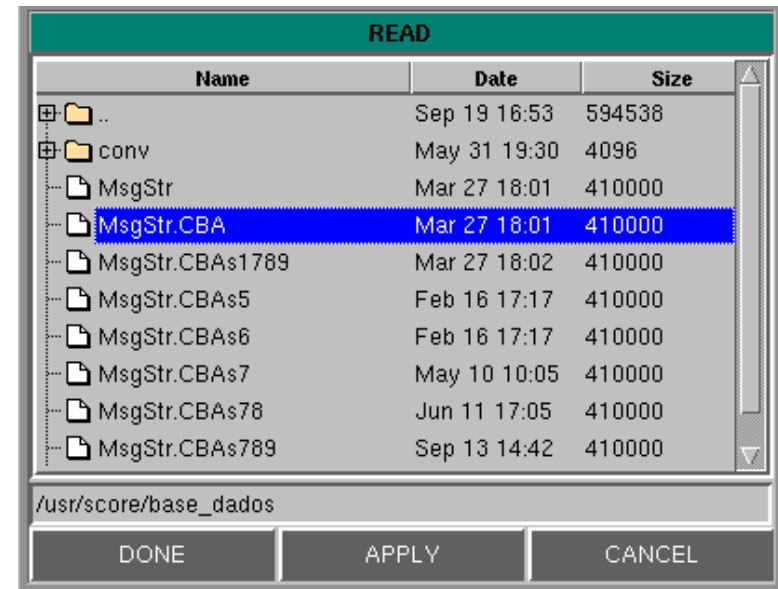
# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings

- **READ** : carregar a base de dados selecionada para memória do editor:

### Arquivos de Bases de Dados

- **Diretório:** [/score/base\\_dados](#)
- **Bases de Dados:**
  - [MsgStr](#): usado pelo Score
  - [MsgStr.CBA](#): reduções II, III e IV
  - [MsgStr.CBAs1789](#): reduções I, V VI e VII



- Características da Tela:
  - **Tree-View**: área para seleção do arquivo de base de dados
  - **Path**: área para seleção da base de dados para configurar
  - **Botões**: **DONE** – Finalizar, **APPLY** – Confirmar, e **CANCEL** – Sair ou Abandonar

# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings

- **WRITE** : salvar a base de dados da memória no disco. A operação é confirmada pela mensagem ao lado



- **GOTO** : posicionar na linha definida pela tela ao lado para alteração da string.



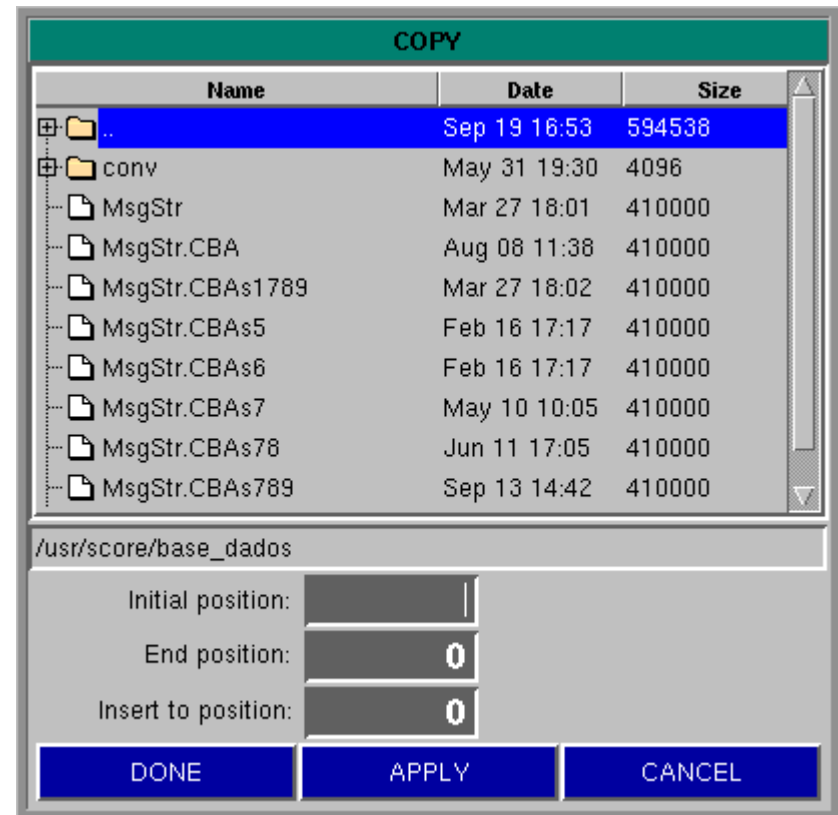
- **SEARCH** : pesquisar a string informada pela tela ao lado



# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings

- **COPY** : copiar uma área da base de dados selecionada pela tela abaixo para memória do editor
- Características da Tela:
  - **Tree-View** : área de seleção do arquivo de base de dados
  - **Path** : área para definição do path do arquivo
  - **Coordenadas da Transferência**:
    - **Inicial position**: posição inicial da área copiada
    - **End position**: posição final da área copiada
    - **Insert to position**: posição inicial que a área será inserida



- **Botões**: **DONE** – Finalizar, **APPLY** – Confirmar e **CANCEL** – Sair ou Abandonar

# Ferramentas de Configuração

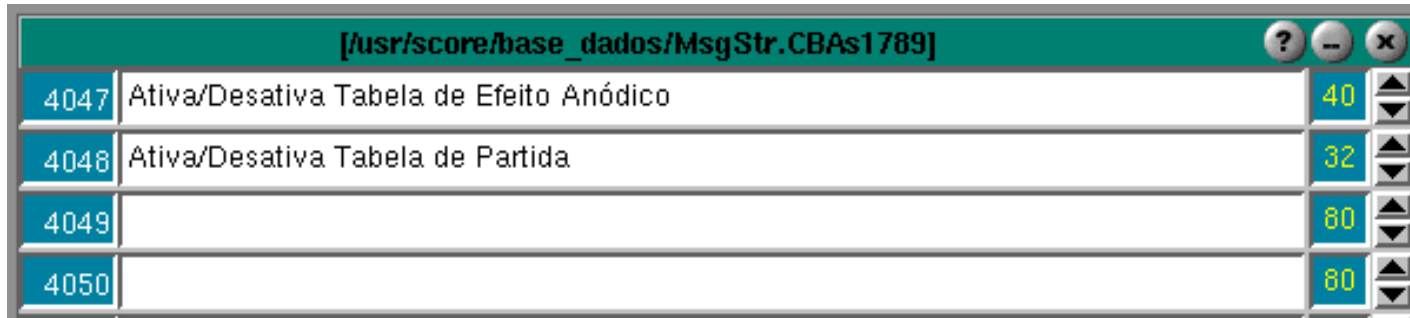
## EditBdStr - Edição da Base de Dados de Strings



- **LOAD** : essa funcionalidade é implementada pelo programa **/score/util/LoadBdStr** executado através do shell do QNX, não está implementada nessa interface  
Esse botão funciona somente quando o programa **EditBdStr** é ativado a partir do diretório **/score/util**
- **QUIT** : sair do editor

# Ferramentas de Configuração

## EditBdStr - Edição da Base de Dados de Strings



The screenshot shows a window titled "[usr/score/base\_dados/MsgStr.CBAs1789]". It contains a table with four rows. Each row has a blue index column on the left, a text field in the center, a blue size column on the right, and a vertical scrollbar on the far right.

Index	String	Size
4047	Ativa/Desativa Tabela de Efeito Anódico	40
4048	Ativa/Desativa Tabela de Partida	32
4049		80
4050		80

### Edição de Strings

- Uma mensagem é formada por 2 atributos da base de dados de strings:
  - **Posição:** posição sequencial da mensagem na base de dados
  - **Mensagem:** conteúdo da string que é exibido na tela ou evento
  - **Tamanho:** tamanho da mensagem, **default = 80**
- A string sempre deve ser salva após a alteração



# Ferramentas de Configuração

---

## Laboratório

- Incluir a chamada da ferramenta EditBdStr na régua de ferramentas do photon:
  - Criar o grupo Score Tools
  - Incluir a chamada do EditBdStrph
- Carregar o editor de strings
  - Carregar a base de strings da redução que está em execução
  - Posicionar na string **620**
  - Incluir a palavra **Novo** nas strings **623** e **627**
  - Salvar as alterações
- Retirar a IHM de supervisão e partir novamente
  - Verificar se a alteração que você fez apareceu na tela de relatórios

## LoadBdStr - Instalação da Base de Dados de Strings

**1º Passo** : Copiar a base de dados de string **PathStr** alterada para o diretório **/score/base\_dados** do micro de controle da redução em execução, onde **PathStr** é :

- **/score/base\_dados/MsgStr.CBA**: reduções II, III e IV
- **/score/base\_dados/MsgStr.CBAs1789**: reduções I, V, VI e VII

**2º Passo** : carregar a base de dados de string atualizada pelo utilitário **LoadBdStr** passando como parâmetro o arquivo **PathStr**, conforme a sintaxe abaixo:

```
# /score/util/LoadBdStr /score/base_dados/PathStr
```

O utilitário **LoadBdStr** copia a base de dados **PathStr** para o arquivo lido pelo sistema **/score/base\_dados/MsgStr** e carrega esse arquivo atualizado na memória principal se o sistema estiver em execução.

# Ferramentas de Configuração

---

## Laboratório

- Carregar a respectiva base de dados de strings do sistema que está em execução no na QNX4\_VM-1 (controle)
  - Copiar a base de dados de strings alterada no laboratório anterior para o diretório `/score/base_dados` do nodo de controle
  - Carregar a base de dados de strings pelo utilitário **LoadBdStr:**  
`# /score/util/LoadBdStr /score/base_dados/PathStr`
- Retirar a IHM de supervisão e partir novamente
  - Verificar se a alteração que você fez apareceu na tela de relatórios

# Ferramentas de Configuração

## DescrEvph – Configuração do Descritor de Eventos

**1º Passo** : carregar o photon

```
# ph
```

**2º Passo** : chamar o editor pela **Barra de Ferramentas** selecionando **Grupo Score Tools → DescrEvPh**, ou via shell pelo comando abaixo:

```
# /score/util/DscrEvph
```

Esse utilitário edita o arquivo `/score/base_dados/dscr_ev`



# Ferramentas de Configuração

## DescrEvph – Configuração do Descritor de Eventos

- A tela do programa é carregada, a qual é formada por 3 áreas distintas:
  - Régua de Botões
  - Configuração do evento
  - Configuração tipo dos 5 parâmetros do evento

Score DescrEvph - Descritor de Eventos

Pesquisar Inicializar Mover Copiar Salvar Instalar Sair

Cod. Evento 0

Mneumônico EVENTO

Mensagem NAO INICIALIZADO!

Grupo Registra, imprime e exhibe

Dispositivo Todos dispositivos

Tipo Evento Cuba Linha

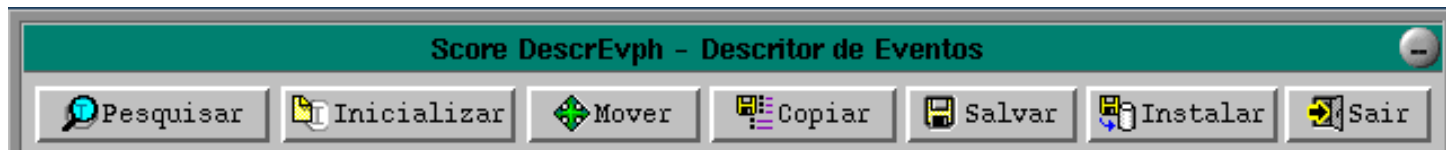
Parâmetros do Evento

Tipo Parâmetro 1	Nenhum
Tipo Parâmetro 2	Nenhum
Tipo Parâmetro 3	Nenhum
Tipo Parâmetro 4	Nenhum
Tipo Parâmetro 5	Nenhum

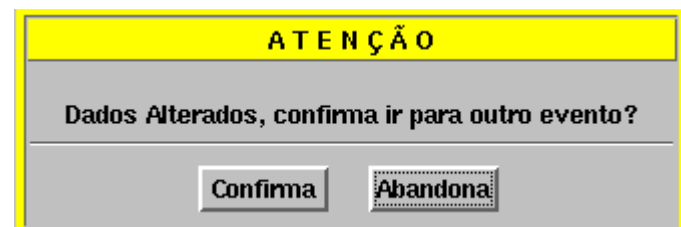
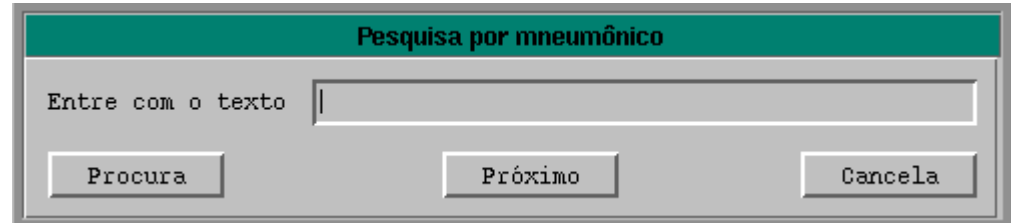
# Ferramentas de Configuração

## DescrEvph – Configuração do Descritor de Eventos

### Régua de Botões



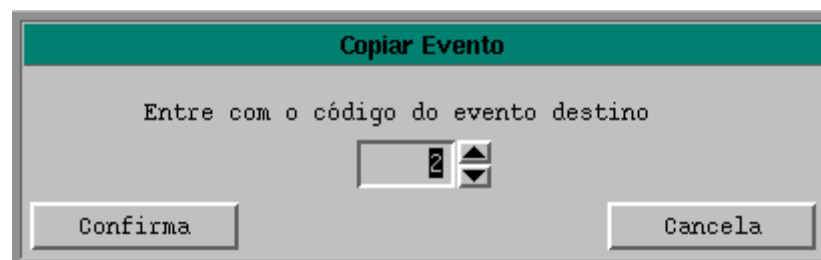
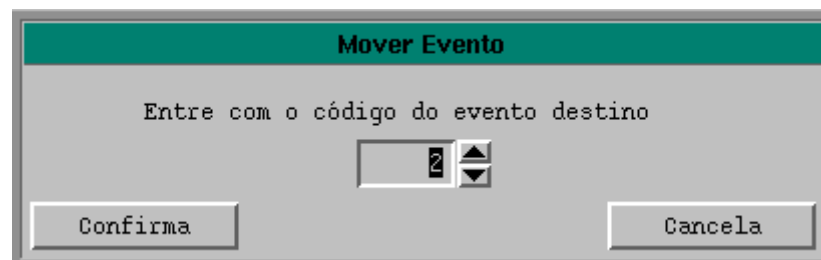
- **Pesquisar** : localizar um evento pelo mneumônico ou mensagem definida na tela abaixo:
  - **Procura** : localiza a primeira ocorrência do texto
  - **Próximo**: localiza próxima ocorrência do texto
  - **Cancela** : sai da tela de pesquisa e encerra
- **Inicializar** : inicializar todos registros de evento com os valores default. Quando essa inicialização é feita, a tela ao lado é exibida toda vez que muda a posição do registro de evento



# Ferramentas de Configuração

## DescrEvph – Configuração do Descritor de Eventos

- **Mover** : move o registro do evento corrente para a posição informada na tela ao lado
- **Copiar** : copia o registro do evento corrente para a posição informada na tela ao lado
- **Salvar** : grava os dados de memória no arquivo `/score/base_dados/descr_ev`
- **Instalar** : grava os dados de memória no arquivo `/score/base_dados/descr_ev` em arquivo e memória do comum. Essa opção não é utilizada pois normalmente instala-se esse arquivo via upgrade nos micros através do utilitário *IniDescrEv*
- **Sair** : sair do programa

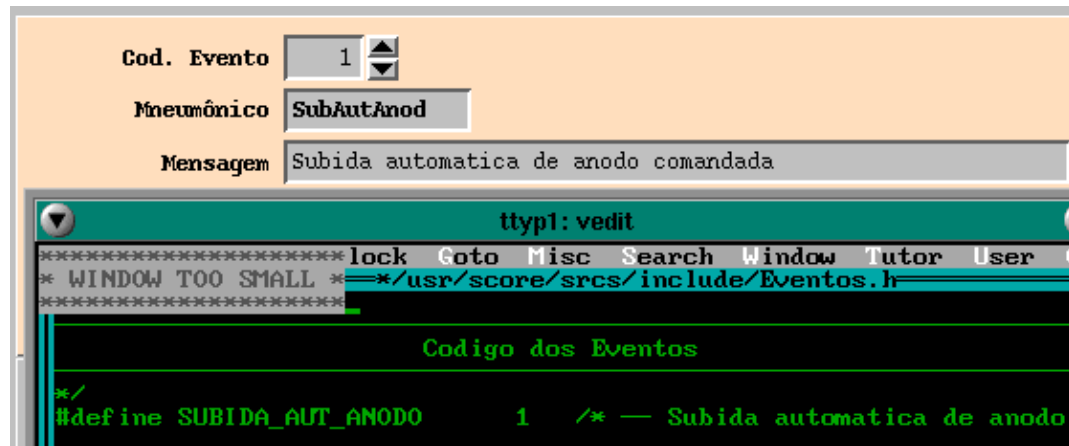


## DescrEvph – Configuração do Descritor de Eventos

### Configuração do Evento

- **Cod. Evento** : posição do evento no arquivo (0 a 200). Essa posição é o mesmo nº do define desse evento do prólogo Eventos.h

A faixa de reservada para eventos do cliente é de 100 a 200.



- **Mneumônico** : código do evento que aparece exibido no relatório de eventos
- **Mensagem** : mensagem do evento exibida na linha de alarmes
- **Grupo** : tratamento do evento de acordo com as opções *Registra Imprime e Exibe*, *Registra Exibe*, *Registra Imprime*, *Apenas Registra*, *Não Registra*



# Ferramentas de Configuração

## DescrEvph – Configuração do Descritor de Eventos

- **Dispositivo** : dispositivos de exibição da mensagem do evento, entre as opções *Todos Dispositivos*, *Exceto Graf Cuba*, *Exceto Ev Inst e Exceto Ev e Graf*
- **Tipo Evento** : indica se é evento de cuba ou linha

### Tipos de Parâmetros do Evento



Parâmetros do Evento	
Tipo Parâmetro 1	Nenhum
Tipo Parâmetro 2	Nenhum
Tipo Parâmetro 3	Nenhum
Tipo Parâmetro 4	Nenhum
Tipo Parâmetro 5	Nenhum

- **Tipo Parâmetro 1 a 5** : configuração do tipo do 1º ao 5º parâmetro do evento de acordo com as opções *Char*, *Int*, *Long*, *Float*, *Str* e *Nenhum* quando o parâmetro não for utilizado.

# Ferramentas de Configuração

---

## Laboratório

- Incluir a chamada da ferramenta EditBdStrph no grupo Score Tools da régua de ferramentas do photon
- Criar um evento com o mneumônico **EvTeste**:
  - Carregar o descritor de eventos **DescrEvph** e criar um evento com o mneumônico **EvTeste** na 1ª posição disponível do descritor de eventos **descr\_ev**
  - Copiar o descritor de eventos alterado para para o diretório **/score/base\_dados** do nodo de controle
- Retirar a IHM de supervisão e partir novamente
  - Verificar se a alteração que você fez apareceu na tela de relatórios

## IniDescrEv – Instalação do Descritor de Eventos

**1º Passo** : instalar o descritor de eventos atualizado no comum pelo utilitário **IniDescrEv** conforme a sintaxe abaixo:

```
# /score/util/IniDescrEv -e -v
```

- O utilitário LoadBdStr copia os dados do arquivo `/score/base_dados/descr_ev` para sua respectiva estrutura de dados do arquivo `/score/base_dados/comum` em disco e na memória principal se o sistema estiver executando

# Ferramentas de Configuração

---

## Laboratório

- Instalar o descritor de eventos alterado na base de dados do sistema que está em execução no na QNX4\_VM-1 (controle)
  - Instalar o descr\_ev já copiado para o diretório /score/base\_dados do nodo de controle pelo utilitário **IniDescrEv**:  
`# /score/util/IniDescrEv -e -v`
- Retirar a IHM de supervisão e partir novamente
  - Verificar se a alteração que você fez apareceu na tela de relatórios

# Ferramentas de Configuração

## DescrRelph – Configuração do Descritor de Relatórios

**1º Passo** : carregar o photon

```
# ph
```

**2º Passo** : chamar o editor pela **Barra de Ferramentas** selecionando **Grupo Score Tools → DescrRelPh**, ou via shell pelo comando abaixo:

```
# /score/util/DscrRelph
```



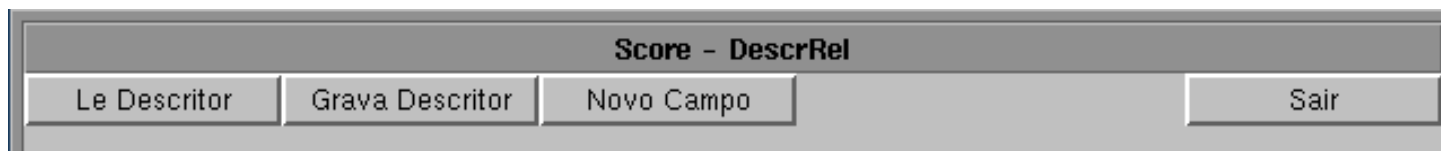
# Ferramentas de Configuração

## DescrRelph – Configuração do Descritor de Relatórios

- A tela do programa é carregada, a qual é formada por 4 áreas distintas:
  - Régua de Botões
  - Configuração do cabeçalho
  - Configuração da área de dados
  - Configuração do rodapé
- A configuração do tamanho das áreas:
  - **Linha Inicial:** 1ª linha da área
  - **N. Linhas:** número de linhas da área configurada
- **Observações:**
  - **Limite Cabeçalho e Rodapé :** 5 linhas
  - **Limite do Relatório:** 20 linhas
  - $\text{LinhaInicial(dados)} = \text{LinhaInicial(Cabeçalho)} + \text{N.Linhas(Cabeçalho)}$
  - $\text{LinhaInicial(Rodapé)} = \text{LinhaInicial(Dados)} + \text{N.Linhas(Dados)}$

## DescrRelph – Configuração do Descritor de Relatórios

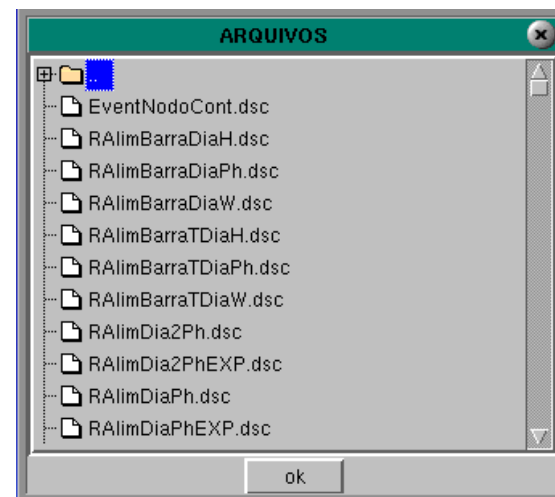
### Régua de Botões



- **Le Descritor** : selecionar o descritor que será carregado. Quando seleciona essa opção, a tela ao lado é exibida com a lista de arquivos do diretório [/score/descr](#).

- **Observações:**

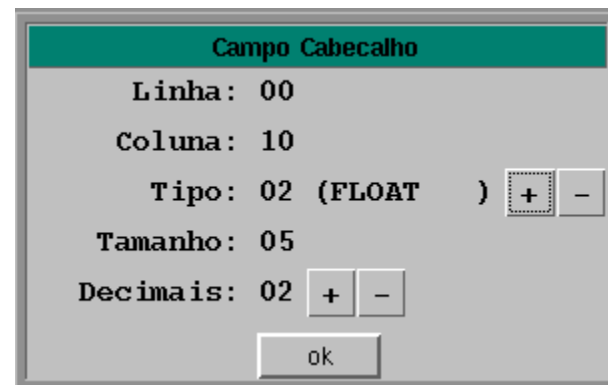
- O nome do descritor sem a extensão [.dsc](#) deve ter o nome do respectivo programa executável.
- Geralmente utilizamos como template um descritor semelhante ao relatório que criamos para agilizar o processo um novo descritor nem salvar com outro nome



# Ferramentas de Configuração

## DescrRelph – Configuração do Descritor de Relatórios

- **Novo Campo** : cria um novo campo no descritor do relatório. O campo é criado através dos seguintes passos:
  1. Preencher com **x** o espaço do novo campo, como na tela ao lado
  2. Marcar o campo com o mouse, conforme a tela ao lado
  3. Selecionar o botão *Novo Campo* para configurar o campo
  4. Configurar o Tipo do campo as opções *INTEIRO* (int), *Float*, *\_ASCII* (char), *LONGO* (long), *UNASIGNED*, *DATA*, *HORA*.





# Ferramentas de Configuração

## DescrRelph – Configuração do Descritor de Relatórios

- **Grava Descritor** : salvar o descritor em disco. Quando é um descritor novo, a tela ao lado é exibida para receber o nome na 1ª vez que o arquivo é salvo.
- **Sair** : sair do programa



# Ferramentas de Configuração

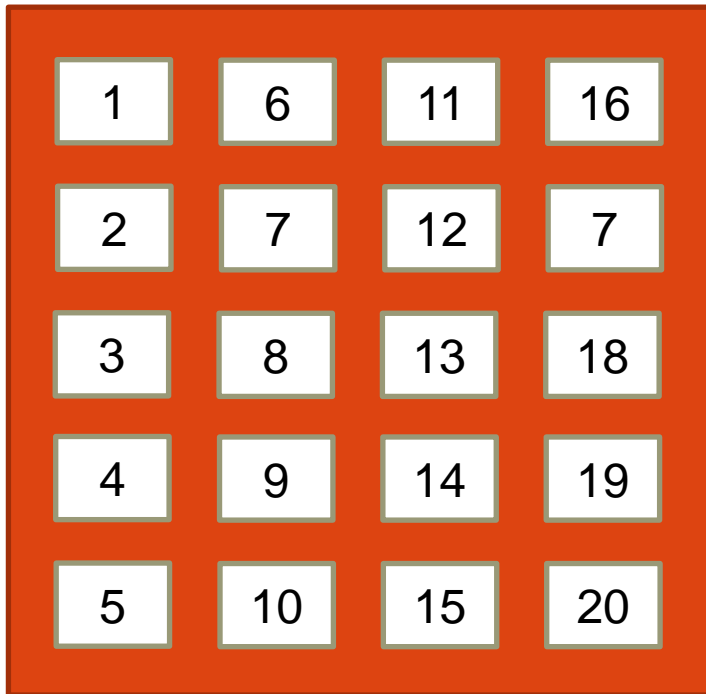
---

## Laboratório

- Incluir a chamada da ferramenta DescrRelph no grupo Score Tools da régua de ferramentas do photon
- Criar um arquivo descritor para o relatório de ciclo de controle do toolkit de desenvolvimento.

# Ferramentas de Configuração

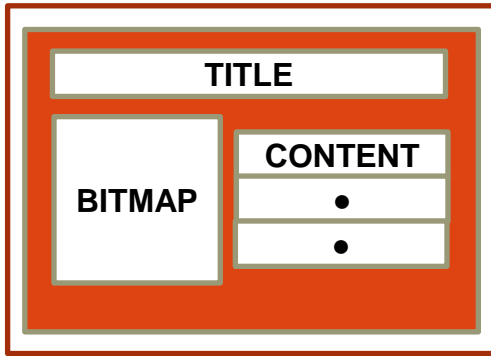
## ConfIHMph – Arquitetura da Tela da IHM



1	6	11	16
2	7	12	7
3	8	13	18
4	9	14	19
5	10	15	20

- **20 Botões:**
  - 5 x 4 (Linhas x Colunas)
  - Configurados via **ConfIHMph**
- **Tipos de Tela:**
  - **RelTela (5)** : relatórios diversos
  - **RelGraf (2)** : relatórios gráficos
  - **Operacao (5)**: opções de operação
  - **Configura (2)**: opções de configuração

## ConfIHMph – Arquitetura da Tela da IHM



- **Atributos Configurados via ConfIHMph**
  - **Bitmap:** ícone do botão
  - **Title:** título do botão (tipo de tela)
  - **Content:** descrição da opção executada

- **Relação entre ConfIHMph x EditBdStrph**
  - **Title e Content:**
    - Configurados na base de dados de strings **MsgStr**
    - **ConfIHMph:** configura na IHM a posição da respectiva mensagem cadastrada na IHM
  - Espaço reservado sequencialmente para botões no **MsgStr**
    - **Botão:** 4 linhas de mensagens (1 Title + 3 Content)
    - **Tela :** 20 botões → 4 x 20 = 80 linhas de mensagens

## ConfHMph – Cálculo da Posição dos Textos do Botão

Posição Inicial $P_t$	
Tela	$P_t$
RelTela1	620
RelTela2	700
RelTela3	780
RelTela4	860
RelTela5	940
RelGraf1	1020
RelGraf2	1100
Operacao1	1180
Operacao2	1260
Operacao3	1340
Operacao4	1420
Operacao5	1500
Configura1	1580
Configura2	1660

- **Cálculo da Posição Inicial do Botão:**

**Formula:**  $P_b = P_t + ((O_b - 1) \times 4) \rightarrow O_b = \{1, 2, 3, \dots, 20\}$

*onde :*  $P_b \rightarrow$  posição inicial do botão (Title)

$P_t \rightarrow$  posição inicial da tela (tabela ao lado)

$O_b \rightarrow$  ordem (posição) do botão na tela

- **Exemplo:** Tela **Operacao1**,  $P_t = 1180$

- **1º Botão:**  $O_b = 1 \rightarrow P_b = 1180 + (1 - 1) \times 4 = 1180$

- **2º Botão:**  $O_b = 2 \rightarrow P_b = 1180 + (2 - 1) \times 4 = 1184$

• • •

- **20º Botão:**  $O_b = 20 \rightarrow P_b = 1180 + (20 - 1) \times 4 = 1256$

# Ferramentas de Configuração

---

## Laboratório

- Abrir a IHM de supervisão do sistema e selecionar a tela de operação
- Abrir a base de dados de strings **MsgStr** pelo **EditBdStr** e localizar os atributos **Title** e **Content** do 1º botão de operação

# Ferramentas de Configuração

## ConfIHMph – Configuração da IHM

**1º Passo** : carregar o photon

```
# ph
```

**2º Passo** : chamar o editor pela **Barra de Ferramentas** selecionando **Grupo Score Tools → ConfIHMPh**, ou via shell pelo comando abaixo:

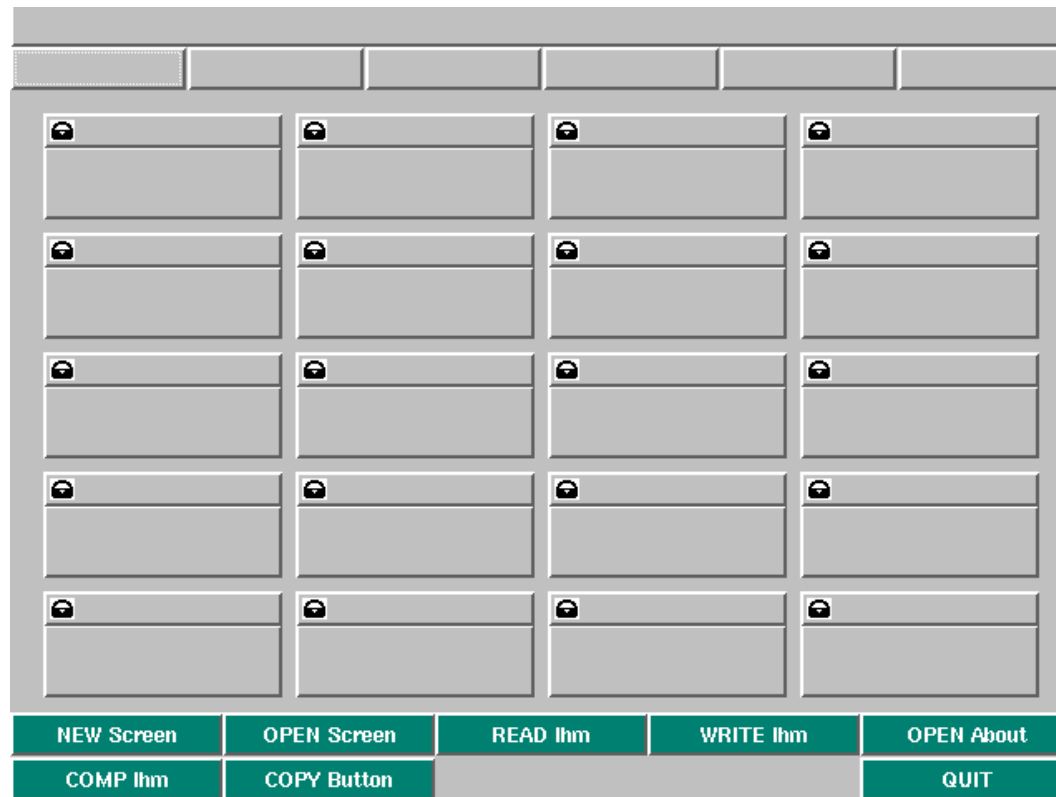
```
# /score/util/ConfIHMph
```



# Ferramentas de Configuração

## ConfIHMph – Configuração da IHM

- A tela do programa é carregada, a qual é formada por 2 áreas distintas:
  - Área da tela carregada
  - Régua de Botões





# Ferramentas de Configuração

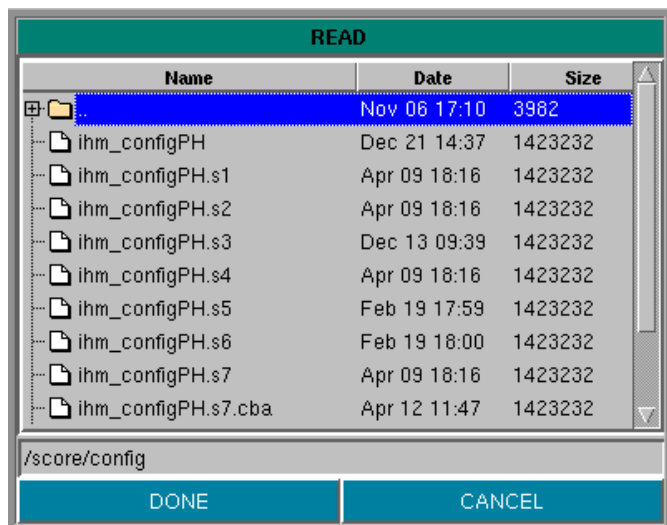
## ConfIHMph – Configuração da IHM

### Régua de Botões

NEW Screen	OPEN Screen	READ Ihm	WRITE Ihm	OPEN About
COMP Ihm	COPY Button			QUIT

- **READ Screen** : selecionar a IHM do diretório */score/config* pela tela abaixo

### IHM x Sala

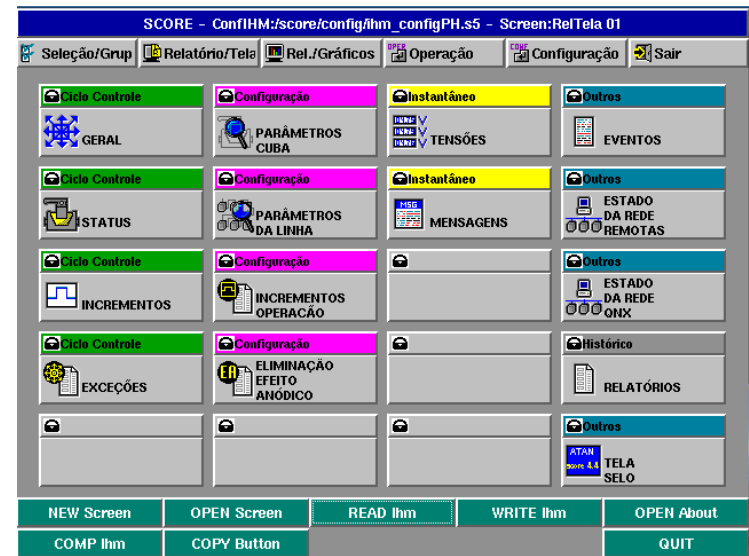


IHM de Supervisão	
Path	Sala
/score/config/ihm_configPH.s1	IHM da sala 125 kA I
/score/config/ihm_configPH.s2	IHM da sala 125 kA II
/score/config/ihm_configPH.s3	IHM da sala 125 kA III
/score/config/ihm_configPH.s4	IHM da sala 125 kA IV
/score/config/ihm_configPH.s5	IHM da sala 125 kA V
/score/config/ihm_configPH.s6	IHM da sala 125 kA VI
/score/config/ihm_configPH.s7	IHM da sala 125 kA VII
/score/config/ihm_configPH.s8	IHM da sala 125 kA VIII
/score/config/ihm_configPH.s9	IHM da sala 125 kA IX

# Ferramentas de Configuração

## ConfIHMph – Configuração da IHM

- A IHM selecionada é carregada conforme a tela ao lado
- **OPEN Screen** : selecionar a tela da IHM que será carregada a partir do combo Box exibido com as seguintes opções:
  - Rel Tela 1
  - Rel Tela 2
  - RelGráfico
  - Operação
  - Configuração



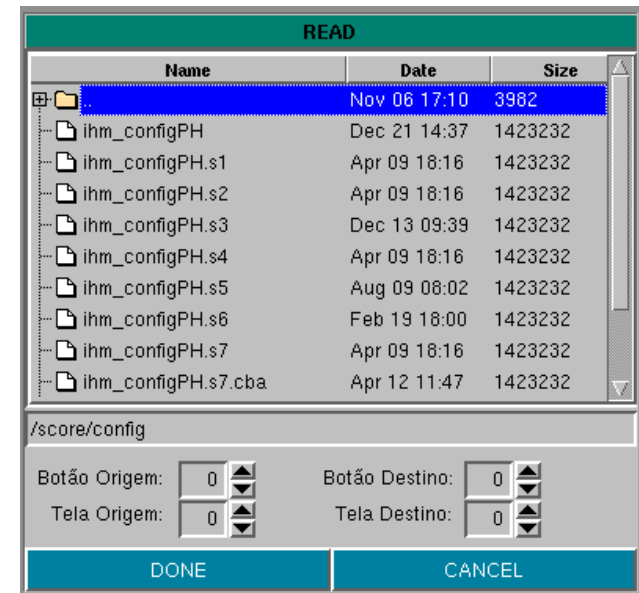
- **NEW Screen** : criar uma nova tela de IHM. Quando essa opção é selecionada um novo gabarito da tela todo em branco é exibido para incluir os botões.
- **WRITE Ihm** : salvar a IHM carregada em disco. Quando essa operação é executada, a mensagem ao lado é exibida informando sua conclusão.



# Ferramentas de Configuração

## ConfIHMph – Configuração da IHM

- **COPY Button** : copia o botão da IHM selecionada (origem) pela tela ao lado para a IHM aberta (destino) de acordo com as coordenadas abaixo:
- **Botão Origem**: botão que será copiado domínio de 0 a 19.
- **Tela Origem**: tela de onde o botão será copiado
- **Botão Destino**: posição da tela para onde o botão será copiado
- **Tela Destino**: tela para onde o botão será copiado



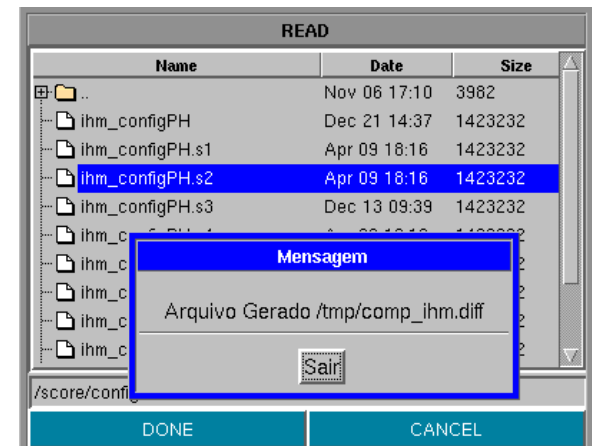
# Ferramentas de Configuração

## ConfIHMph – Configuração da IHM

- A identificação das telas de IHM variam de uma sala para a outra. Na tabela ao lado podemos ver um exemplo a lista de telas da IHM da sala I

Telas da IHM			
Nº	Tela	Nº	Tela
0	RelTela 01	5	RelTela 03
1	RelTela 02	6	ManutScore
2	RelGrafico	7	Operacao2
3	Operacao	8	OperacaoMoveI
4	Configuracao	9	RelTela04

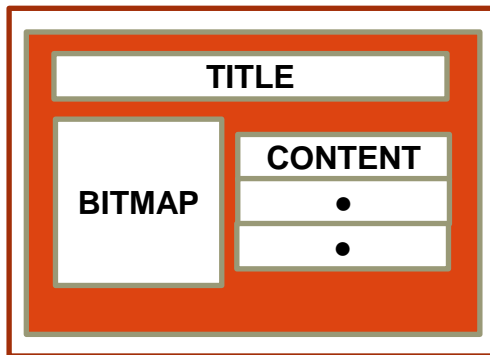
- COMP Ihm** : comparar a IHM em memória com a IHM selecionada na tela ao lado. O resultado da comparação é registrado em arquivo cujo *path* é informado pela mensagem ao lado.



# Ferramentas de Configuração

## ConfIHMph – Configuração da Chamada do Botão

- **1º Passo** : seleccionar a IHM desejada e carregar na memória através do botão **READ Ihm**.
- **2º Passo** : seleccionar a tela desejada e carregar na memória através do botão **OPEN Screen**.
- **3º Passo** : seleccionar o botão e configurar seu lay-out pelos campos **TITLE**, **CONTENT**, e **BITMAP** conforme gabarito abaixo.



Interface de configuração do botão. O título da janela é "BUTTON: 0 SCREEN: 3". Os campos incluem:

- SCREEN**: Fill:  TOP:  Text:
- Name**: Operacao
- TITLE**: Index text: 1420 Fill:  Text:
- Text**: Operação
- CONTENT**: Index text: 1421 Fill:  Text:
- Text**: MOVIMENTAÇÃO
- TYPE**: Disable Menu **Task** Task+Web Action About
- PASSWORD**: Level: 25
- BITMAP**: /score/icon2/cuba\_anod.bmp

Botões de ação: DONE APPLY CANCEL

# Ferramentas de Configuração

## ConfHMph – Configuração da Chamada do Botão

- **4º Passo** : definir nível de senha de acesso pelo campo **PASSWORD** de 0 a 255. Se o **Level** é maior que 0, a exibição do ícone de cadeado é habilitada. Geralmente essa configuração segue o seguinte padrão:
  - **Nível 25**: programas de operação
  - **Nível 50**: módulo histórico
  - **Nível 100**: módulo de configuração, exceto cadastro de usuários
  - **Nível 255**: Hard-Copy, cadastro de usuários e sair do Score
- **5º Passo** : definir o tipo de ação do botão entre as opções **Disable**, **Menu**, **Task** e **Action**.
  - **Disable**: desabilita o botão
  - **Menu**: exibe o menu selecionado no combo-box abaixo da régua de botões
  - **Task**: configura a criação do programa



A screenshot of a configuration window showing a 'PASSWORD' label and a 'Level:' field with the value '25'.



A screenshot of a configuration window showing a 'TYPE' label and a row of buttons: 'Disable', 'Menu', 'Task', 'Task+Web', 'Action', and 'About'. The 'Disable' button is highlighted with a green background.



A screenshot of a configuration window showing a 'TYPE' label and a row of buttons: 'Disable', 'Menu', 'Task', 'Task+Web', 'Action', and 'About'. The 'Menu' button is highlighted with a green background. Below the buttons is a dropdown menu showing 'RelTela 01'.



A screenshot of a configuration window showing a 'TYPE' label and a row of buttons: 'Disable', 'Menu', 'Task', 'Task+Web', 'Action', and 'About'. The 'Task' button is highlighted with a green background.

# Ferramentas de Configuração

## ConfIHMph – Configuração da Chamada do Botão

- **Action**: executa a ação selecionada no combo-box entre as opções:
  - **Pots Selection**: exibe cubas selecionadas antes da ação
  - **End System**: sai da IHM de supervisão e encerra sua execução
- **6º Passo**: configurar os parâmetros de criação do programa quando for tipo **Task**.

A screenshot of a configuration window titled 'TASK'. The window has a green header bar. Below the header, there are several sections. The first section contains checkboxes for 'Write:', 'Export:', 'View Args:', and 'Help:', along with 'Priority: 10', 'Spaw Flags: 0', and 'Args: 4'. The second section contains a 'Path+Task Name:' field with the value '/score/exec/AtivaOperacaoPH'. The third section contains 'Page Arg: 1', 'Task Arg: TEXT STRING FIXED HID', and 'Ihm Arg: NOT INITIALIZED'. The fourth section contains 'Label Index: -1'. The fifth section contains 'Default Index:'. The sixth section contains 'Default Value: OpeSPassoTabIncPH'. The bottom section is a blue area with labels for 'Type List:', 'Items List:', 'Titles:', 'Title1 Index:', 'Title2 Index:', 'List1 Index:', 'List2 Index:', 'List1 Value:', and 'List2 Value:'.

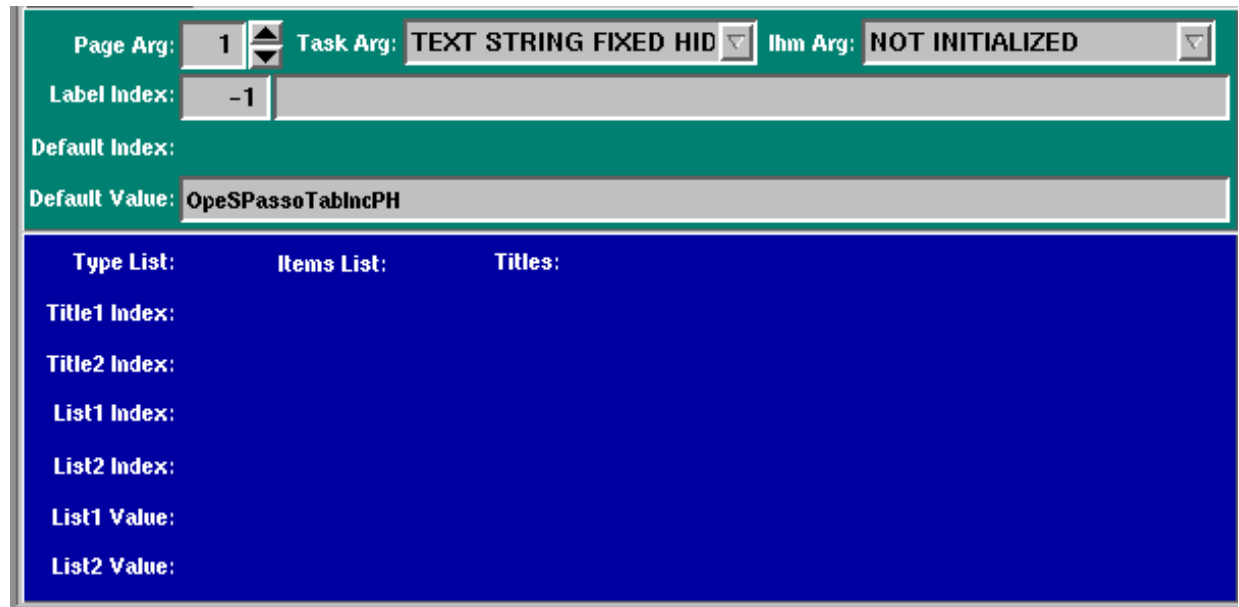
# Ferramentas de Configuração

## ConfHMph – Configuração da Ativação do Programa

- A ativação dos programas configurada a através de 3 áreas da tela:
  - Interface**: flags de interface
  - Criação Programa**: flags e *path* de criação



- Argumentos**:  
parâmetros  
passados como  
argumentos na  
criação do  
programa

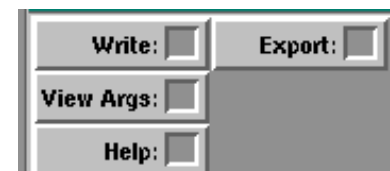




# Ferramentas de Configuração

## ConfHMph – Configuração da Ativação - Interface

- **Interface**: a configuração de chamada da interface é feita através dos seguintes flags:
  - **Write** : grava os parâmetros como default para próxima chamada
  - **View Args** : exhibe os argumentos antes de chamar o programa
  - **Export** : pede confirmação antes de ativar o programa
  - **Help** : habilita arquivo de ajuda do programa *NomeProg*, cujo *path* será */score/help/NomeProg.hlp*



# Ferramentas de Configuração

## ConfIHMph – Configuração da Ativação - Criação

- **Criação Programa** : a configuração dos parâmetros de criação é feita pelas seguintes opções:
  - **Nome + Task Name** : *path* do programa que será criado.

### **Observação :**

O path dos programas de operação é configurado conforme a tela abaixo com o *path* `/score/exec/AtivaOperacaoPH` e o nome do programa `NomeProg` sem *path* no campo ***Default Value***.



The screenshot displays the configuration window for creating a program. It features several input fields and checkboxes. The 'Path+Task Name' field is set to '/score/exec/AtivaOperacaoPH'. The 'Default Value' field is set to 'OpeSPassoTabIncPH'. Other visible settings include 'Priority' at 10, 'Spaw Flags' at 0, 'Args' at 4, 'Page Arg' at 1, 'Task Arg' set to 'TEXT STRING FIXED HID', and 'Ihm Arg' set to 'NOT INITIALIZED'.

Write: <input type="checkbox"/>	Export: <input type="checkbox"/>	Priority: 10	Path+Task Name: /score/exec/AtivaOperacaoPH
View Args: <input type="checkbox"/>		Spaw Flags: 0	
Help: <input type="checkbox"/>		Args: 4	
Page Arg: 1	Task Arg: TEXT STRING FIXED HID	Ihm Arg: NOT INITIALIZED	
Label Index: -1			
Default Index:			
Default Value: OpeSPassoTabIncPH			

No exemplo de tela vemos a criação do programa `OpeSPassoTabIncPH`

## ConfIHMph – Configuração da Ativação - Criação

- **Prioridade** : prioridade de execução do programa, geralmente executam com prioridade 10
- **Spawn Flags** : flags de criação da tarefa passados no comando *spawn*
- **Num Args** : número de argumentos relacionados à criação do programa que foram configurados na IHM.  
Importante destacar que esse parâmetro não tem nenhuma relação com a variável *argc* utilizada no programa em C.

# Ferramentas de Configuração

## ConfHMph – Configuração da Ativação - Criação

- **Argumentos** : a configuração dos ***n*** argumentos definidos em ***Args*** é feita pelas seguintes opções:
  - **Page Arg** : seleciona a pagina de configuração do argumento
  - **Task Arg** : tipo do argumento passado para o programa

Arguments Type Defined	
TASK ARG	Description
ALGORITM	Algoritmo definido na tela de pré-seleções
DATE END	Data Final definida na tela de pré-seleções
DATE INITIAL	Data Inicial definida na tela de pré-seleções
DAY EXCLUDED	Lista de datas excluídas definida na tela de pré-seleções
LIST	Tela gráfica com botões de opções associado aos atributos de configuração de lista
OPERATOR	Identificação do operador passada durante o pedido de senha para o acesso à opção desejada.
POTS EXCLUDED	Lista de cubas excluídas definida na tela de pré-seleções

# Ferramentas de Configuração

## ConfHMph – Configuração da Ativação - Criação

Continuação dos tipos de argumentos **Task Arg...**

Arguments Type Defined	
TASK ARG	Description
POTS KEYBOARD	Faixa de cubas definida via teclado acionado no rodapé da tela de chamada da opção.
POTS SELECTION	Faixa de cubas definida na tela de pré-seleções
POTS STRING	Faixa de cubas definida no formato de string.
REDUCTION	Número da redução do projeto
SHIFT	Turno definido na tela de pré-seleções
TEXT STRING	Tipo alfanumérico definido através de teclado alfabético que é vinculado aos campos <b>Label Index</b> e <b>Default Index</b>
TEXT STRING FIXED	Permite a passagem de um parâmetro interno fixo de execução do programa.
TEXT STRING FIXED HIDDEN	Permite a passagem de um parâmetro interno fixo de execução do programa oculto do usuário. Geralmente esse tipo é utilizado para passar argumentos para o programa que está sendo ativado.

# Ferramentas de Configuração

## ConfIHMph – Configuração da Ativação - Criação

- IHM Arg : tipo do argumento passado para a IHM conforme tabela abaixo

Arguments Type Defined	
IHM ARG	Description
CONFIRMS QUESTION	Exibe pergunta definida no campo <b>Default Index</b> para confirmar a execução, cuja resposta deve ser sempre <b>Confirma</b> ou <b>Abandona</b> .
TITLE	Título exibido no topo da tela de chamada da opção, que é definido no campo <b>Default Index</b> e não é passado para o programa acionado.
HELP	Determina se a opção terá arquivo de ajuda. Se essa opção estiver selecionada o arquivo de ajuda deverá ter o mesmo nome do módulo executável com a extensão “ <b>.hlp</b> ”, ficando no diretório <b>/score/help</b>

- Label Index : nome do campo da tela
- Default Index e Default Value : esses itens representam o valor do argumento, onde **Default Index** é a identificação do argumento exibido ao usuário na tela e **Default Value** o respectivo valor repassado ao programa como argumento e oculto ao usuário

# Ferramentas de Configuração

## ConfIHMph – Configuração da Ativação – Criação (Lista)

- **Task Arg: List** : o argumento tipo lista deve ter sempre Type List : 0 para possibilitar a configuração dos demais parâmetros da lista, pois Type List : 1 especifica a lista utilizada exclusivamente no relatório de eventos montada através do programa da IHM.

A lista é configura através dos parâmetros da tela abaixo:

Type List: 0 Items List: 1 Titles: 1

Title1 Index: 2081 Direção

Title2 Index:

List1 Index: 2080 Subir,Descer

List2 Index:

List1 Value: 1,2

List2 Value:

- **Items List** : nº de itens da lista que o campo vai receber (máximo 15)
- **Titles** : nº de áreas da tela (máximo 2), onde cada área equivale a 1 lista.
- **Title1 Index/Title2 Index** : título da 1ª e 2ª área respectivamente
- **List1 Value/List2 Value** : respectivos tags dos botões passados como **argv[ ]** para os programas de operação de botões da 1ª e 2ª área (separados por virgula)

# Ferramentas de Configuração

---

## Resumo de Configuração da Chamada

- Resumo do procedimento de configuração da chamada:
  1. Selecionar a tela e o botão a configurar
  2. Configurar as condições de chamada da interface
  3. Configurar a chamada do programa
  4. Configurar os atributos dos argumentos de chamada



# Ferramentas de Configuração

---

## Laboratório

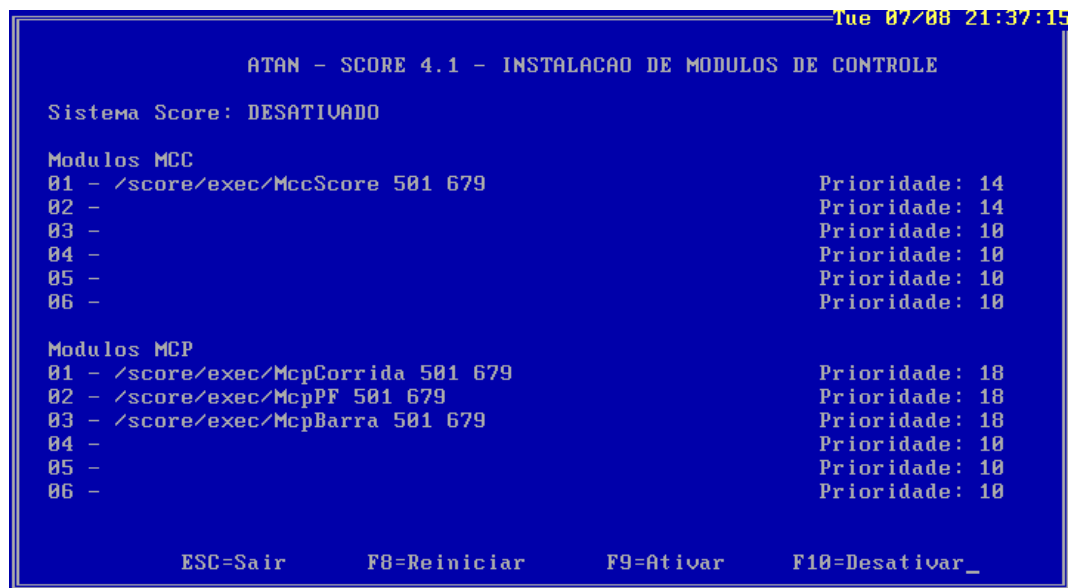
- Incluir a chamada da ferramenta ConflHMph no grupo Score Tools da régua de ferramentas do photon
- Incluir um botão de chamada do programa de operação de exemplo do toolkit de desenvolvimento

# Ferramentas de Instalação

## InstalaCtrl - Configuração de Tarefas Residentes

- Chamar o configurador pelo comando abaixo:

# /score/util/InstalaCtrl



```
Tue 07/08 21:37:15
ATAN - SCORE 4.1 - INSTALACAO DE MODULOS DE CONTROLE

Sistema Score: DESATIVADO

Modulos MCC
01 - /score/exec/MccScore 501 679          Prioridade: 14
02 -                                     Prioridade: 14
03 -                                     Prioridade: 10
04 -                                     Prioridade: 10
05 -                                     Prioridade: 10
06 -                                     Prioridade: 10

Modulos MCP
01 - /score/exec/McpCorrida 501 679        Prioridade: 18
02 - /score/exec/McpPF 501 679            Prioridade: 18
03 - /score/exec/McpBarra 501 679         Prioridade: 18
04 -                                     Prioridade: 10
05 -                                     Prioridade: 10
06 -                                     Prioridade: 10

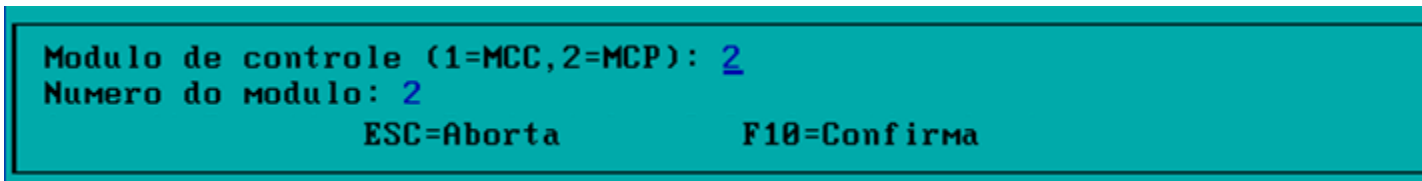
ESC=Sair      F8=Reiniciar    F9=Ativar     F10=Desativar_
```

- Tela composta por 3 áreas:
  - Módulos MCC:** área para configurar a partida de 6 tarefas MCC
  - Módulos MCP:** área para configurar a partida de 6 tarefas MCP
  - Rodapé:** régua de botões para alterar o cadastro e sair

# Ferramentas de Instalação

## InstalaCtrl - Configuração de Tarefas Residentes

- **F8 = Reiniciar** : selecionar essa opção para reiniciar uma tarefa pela tela abaixo:



The screenshot shows a teal-colored terminal window with black text. The text displays the configuration process for a resident task. It shows the selection of the control module (MCC or MCP) and the module number. The current selection is module 2 (MCP). The screen also indicates the function keys for aborting (ESC) and confirming (F10) the configuration.

```
Modulo de controle (1=MCC,2=MCP): 2
Numero do modulo: 2
                        ESC=Aborta      F10=Confirma
```

- Configuração dos parâmetros de reinicialização:
  - **Modulo de controle (1=MCC, 2=MCP):** seleciona **1** para programa de ciclo de controle (MCC) ou **2** para ciclo de leitura (MCP)
  - **Numero do módulo:** seleciona de **1** a **6** para informar a posição do programa na base de dados
  - Confirmar pelo botão **F10=Confirma** ou abandonar por **ESC=Aborta**

# Ferramentas de Instalação

## InstalaCtrl - Configuração de Tarefas Residentes

- **F9 = Ativar** : selecionar a opção para instalar ou ativar o programa pela tela abaixo:

```
Modulo de controle (1=MCC,2=MCP): 2
Numero do modulo: 2
Nome do modulo (com path):
Prioridade: 18
Acao (1=Ativar e Instalar,2=So instalar): 1
                        ESC=Aborta          F10=Confirma
```

- Configuração dos parâmetros de instalação:
  - **Modulo de controle (1=MCC, 2=MCP):** seleciona **1** para programa de ciclo de controle (MCC) ou **2** para ciclo de leitura (MCP)
  - **Numero do módulo:** seleciona de **1** a **6** para informar a posição do programa na base de dados
  - **Nome do módulo (com path):** *path* completo do programa instalado  
**Esse path errado desativa o sistema imediatamente na partida**
  - **Prioridade:** **14** → MCP e **18** → MCC
  - **Ação :** **1** → instalar na base de dados e ativar ou **2** → somente instalar
- Confirmar pelo botão **F10=Confirma** ou abandonar por **ESC=Aborta**

# Ferramentas de Instalação

## InstalaCtrl - Configuração de Tarefas Residentes

- **F10 = Desativar** : selecionar essa opção para desinstalar ou desativar o programa pela tela abaixo:

```
Modulo de controle (1=MCC,2=MCP): 2
Numero do modulo: 2
Acao (1=Desativar e Desinstalar,2=So desativar): 1
                        ESC=Aborta          F10=Confirma
```

- Configuração dos parâmetros de desinstalação:
  - **Modulo de controle (1=MCC, 2=MCP)**: seleciona **1** para programa de ciclo de controle (MCC) ou **2** para ciclo de leitura (MCP)
  - **Numero do módulo**: seleciona de **1** a **6** para informar a posição do programa na base de dados
  - **Ação** : **1** → desativar e desinstalar da base de dados ou **2** → somente desativar
- Confirmar pelo botão **F10=Confirma** ou abandonar por **ESC=Aborta**
- **ESC=Sair** : selecionar essa opção para sair do configurador

ESC=Sair

F8=Reiniciar

F9=Ativar

F10=Desativar\_

# Ferramentas de Instalação

---

## Laboratório

- Alterar o path do programa **Mcp** e carregar a alteração na base de dados
- Verificar o que acontece na partida do sistema

# Ferramentas de Instalação

## selred - Seleção da Base de Dados de Simulação

- A plataforma de desenvolvimento pode ser executada para qualquer redução, cuja base de dados deve ser previamente selecionada.
- As bases de dados de simulação das reduções já vem configuradas com os drivers de demonstração para testar o sistema. Essas bases de dados foram organizadas nos diretórios do quadro abaixo.

Diretório de Base de Dados de Simulação	
Diretório	Conteúdo
/score/base_dados_r1	Base de dados de redução I
/score/base_dados_r2	Base de dados de redução II
/score/base_dados_r3	Base de dados de redução III
/score/base_dados_r4	Base de dados de redução IV
/score/base_dados_r7	Base de dados de redução V
/score/base_dados_r8	Base de dados de redução VI
/score/base_dados_r9	Base de dados de redução VII

## selred - Seleção da Base de Dados de Simulação

- A base de dados da redução que será testada é selecionada pelo comando **selred** passando como parâmetro o número da redução:
  - # /score/selred Redução**
    - ▶ deve ser executado com o sistema desativado
  - # selred 3**
    - ▶ configura sistema para rodar redução III



# Ferramentas de Instalação

---

## Laboratório

- Desativar o sistema e selecionar a base de dados da redução III
- Partir o sistema e verificar se o sistema partiu corretamente

## Configuração da Base de Dados de Demonstração

- A base de dados do runtime do sistema nos micros de controle é configurada com os drivers de dispositivos de hardware.
- Os drivers precisam ser substituídos por outros de simulação para rodar o sistema na plataforma de desenvolvimento.
- A configuração da base de dados para rodar como demonstração (simulação) é feita através do comando *InstalaDemo* do diretório /score/util conforme a sintaxe abaixo:
  - # **InstalaDemo [-o]**      ► o parâmetro `-o` configura os nodos 3 e 4 como nodos de operação

# Ferramentas de Instalação

---

## Laboratório

- Desativar o sistema e instalar um backup de produção da base de dados da redução III
- Partir o sistema e verificar o que acontece
- Executar o programa **InstalaDemo**
- Partir o sistema e verificar se funcionou normalmente

# Ferramentas de Compilação

---

## Compilação do Sistema

- A compilação do sistema é feita através do compilador **wcc** e as bibliotecas geradas pelo **wlib**. Entretanto, no Score esses recursos , assim como outros foram encapsulados em alguns comandos, que podem ser vistos abaixo:
  - # **GeraScore**      ► compila o sistema passando pelos diretórios. Se houver algum erro de sintaxe a compilação é interrompida
  - # **GeraTudo**      ► compila o sistema passando pelos diretórios. Se houver algum erro de sintaxe a compilação passa para o diretório seguinte
  - # **apaga\_objs**    ► apaga todos arquivos objeto .o para forçar a compilação do sistema completo

# Ferramentas de Instalação

---

## Laboratório

- Compilar o sistema completo

Apresentação

Ferramentas de Desenvolvimento

**Arquitetura do Sistema**

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Arquitetura do Sistema

## Principais Tarefas do Kernel do Sistema

Tarefa	Função
<b>AdminSema</b>	Gerenciamento de semáforos de acesso à base de dados
<b>BdScore</b>	Leitura/gravação da base de dados via rede.
<b>Drvsad</b>	Driver de comunicação com a ATN 1.4
<b>EnvRemota</b>	Envia mensagens de comando para as remotas via rede Echelon
<b>Eventos</b>	Gerenciamento de eventos do sistema.
<b>EventosMsg</b>	Indicar os alarmes de eventos na linha de alarmes da IHM
<b>IhmConsole</b>	IHM da console do micro de controle.
<b>Log</b>	Gerenciamento de operações com arquivos de logs do módulo histórico
<b>LogSon</b>	Instanciada a cada log criado para gerenciar a atualização do respectivo arquivo de log
<b>RecChaves</b>	Reconhecer e tratar as sinalizações de mudança de chaves do painel da cuba
<b>RecRemota</b>	Receber as mensagens das remotas pela rede Echelon e envia-as ao driver da PCLTA
<b>RedeAtn7</b>	Driver de comunicação com a placa PCLTA
<b>SalvaBd</b>	Salvar a área comum, log, eventos e históricos da memória principal para HD e Hot-Standby
<b>Score</b>	Partir o sistema Score e criar suas principais tarefas residentes
<b>TabRelGraf</b>	Criar a estrutura de alarmes, gráficos e mini-gráficos o startup e gerenciar sua atualização a cada ciclo de controle
<b>WatchCtrS</b>	Verificar para o Hot-Standby se o controle está ativo no outro micro via cabo serial
<b>Watchdog</b>	Verificar a integridade do sistema e comandar a transferência para o Hot-Standby

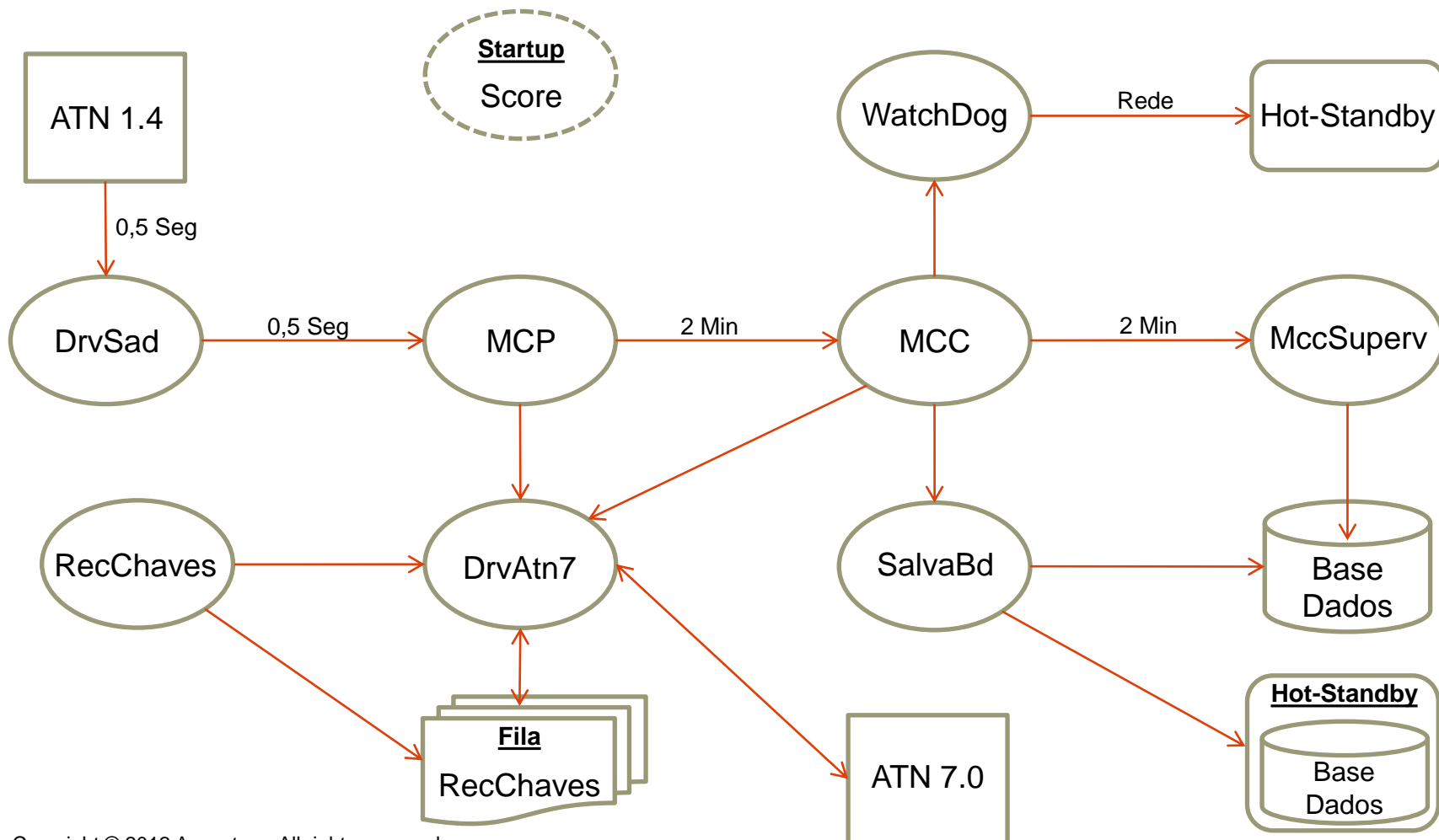
## Principais Tarefas de Controle de Processo

Tarefa	Função
<b>Mcc</b>	Implementar o ciclo de controle ajustando a resistência da cuba e ativando as demais tarefas desse ciclo a cada 2 minutos
<b>Mcp</b>	Implementar o ciclo de leitura, ativar as demais tarefas desse ciclo e a cada 240 ciclos de leitura (2 minutos) ativar o ciclo de controle
<b>McpCorrida</b>	Tratar a corrida de metal das cubas no ciclo de leitura.
<b>TrataEa</b>	Tratar o efeito anódico da cuba.



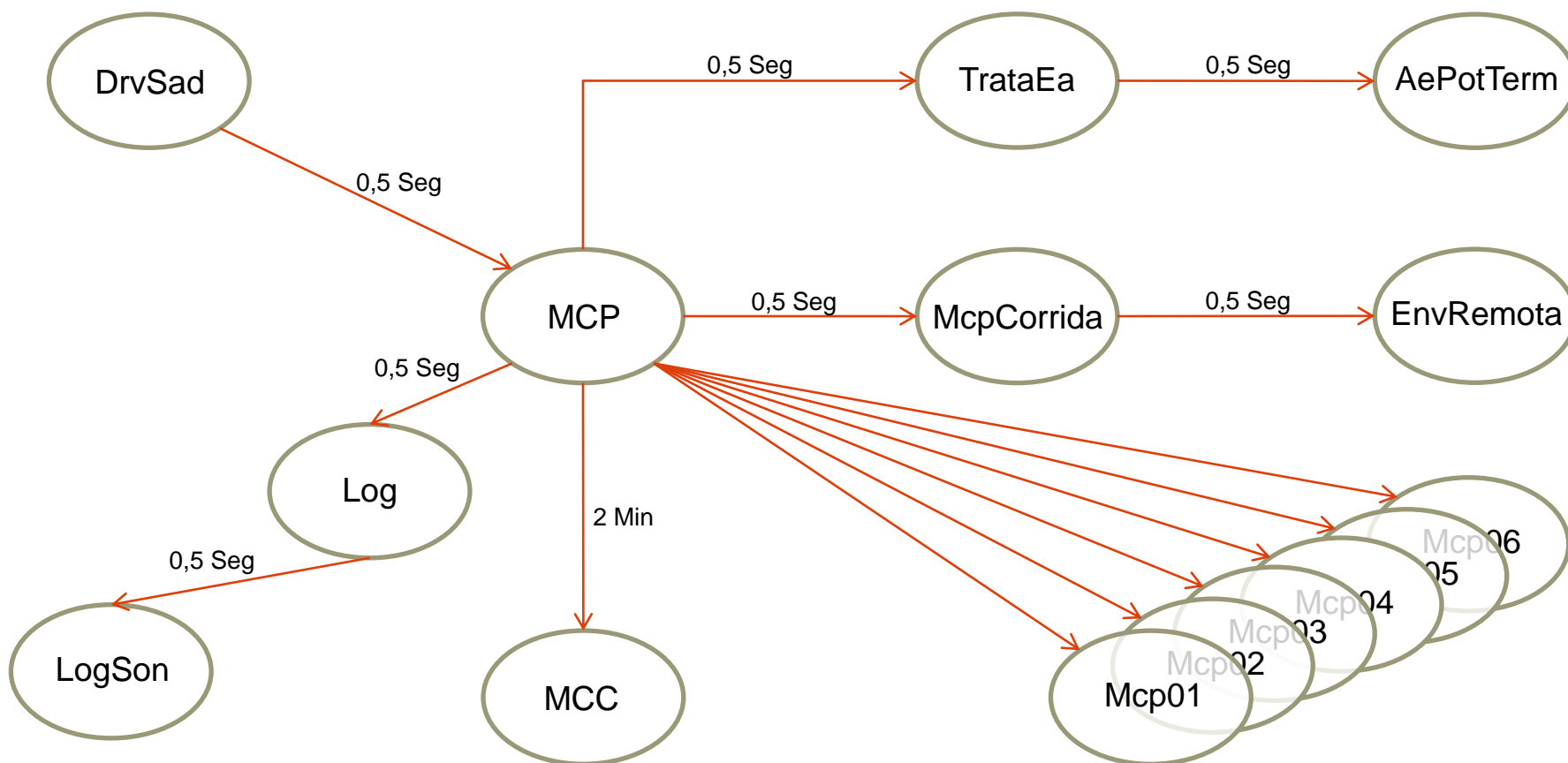
# Arquitetura do Sistema

## Diagrama de Comunicação – Visão Geral



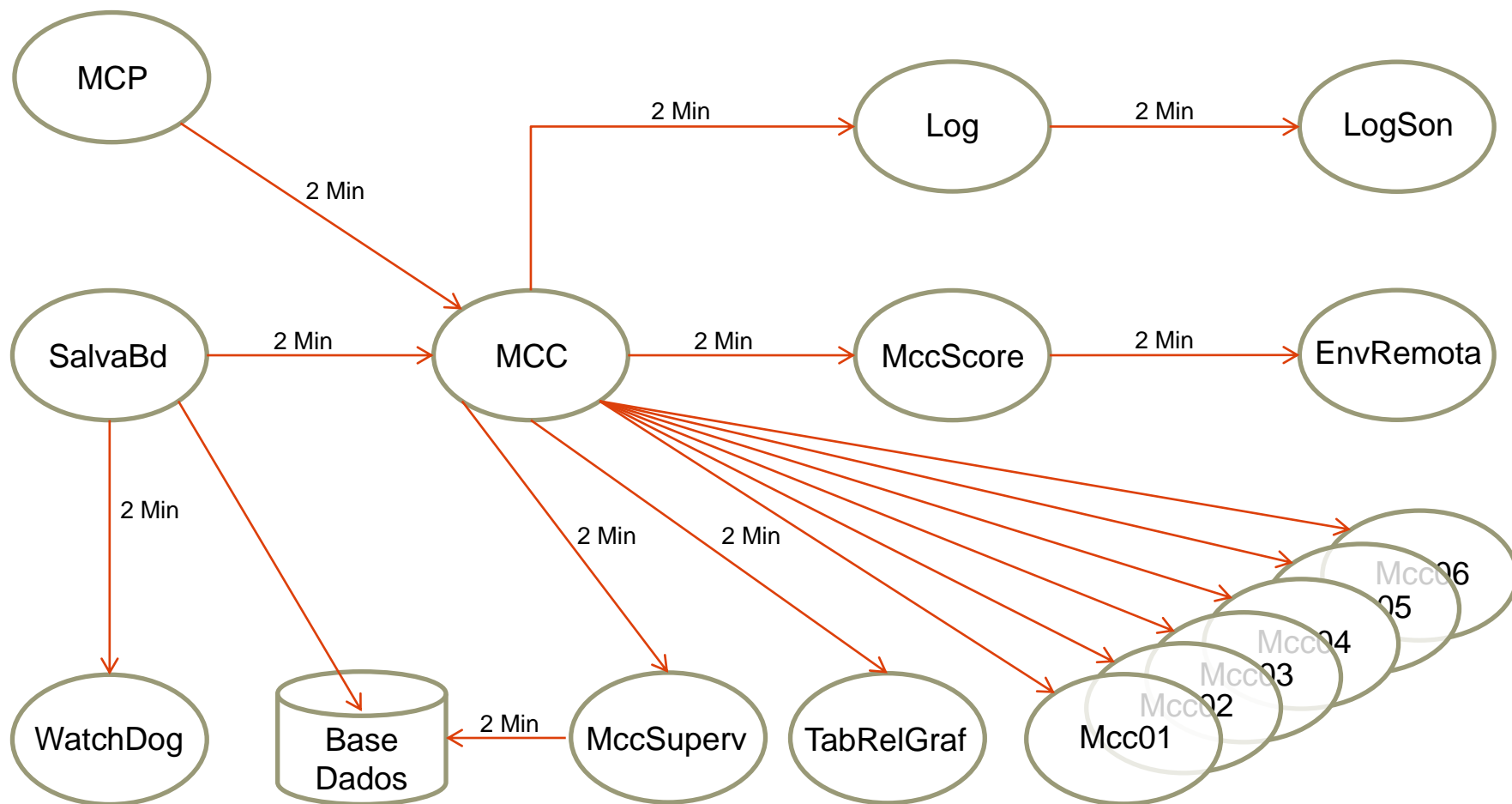
# Arquitetura do Sistema

## Diagrama do Ciclo de Leitura – MCP (0,5 Seg)



# Arquitetura do Sistema

## Diagrama do Ciclo de Controle – MCC (2 Min)



# Arquitetura do Sistema

---

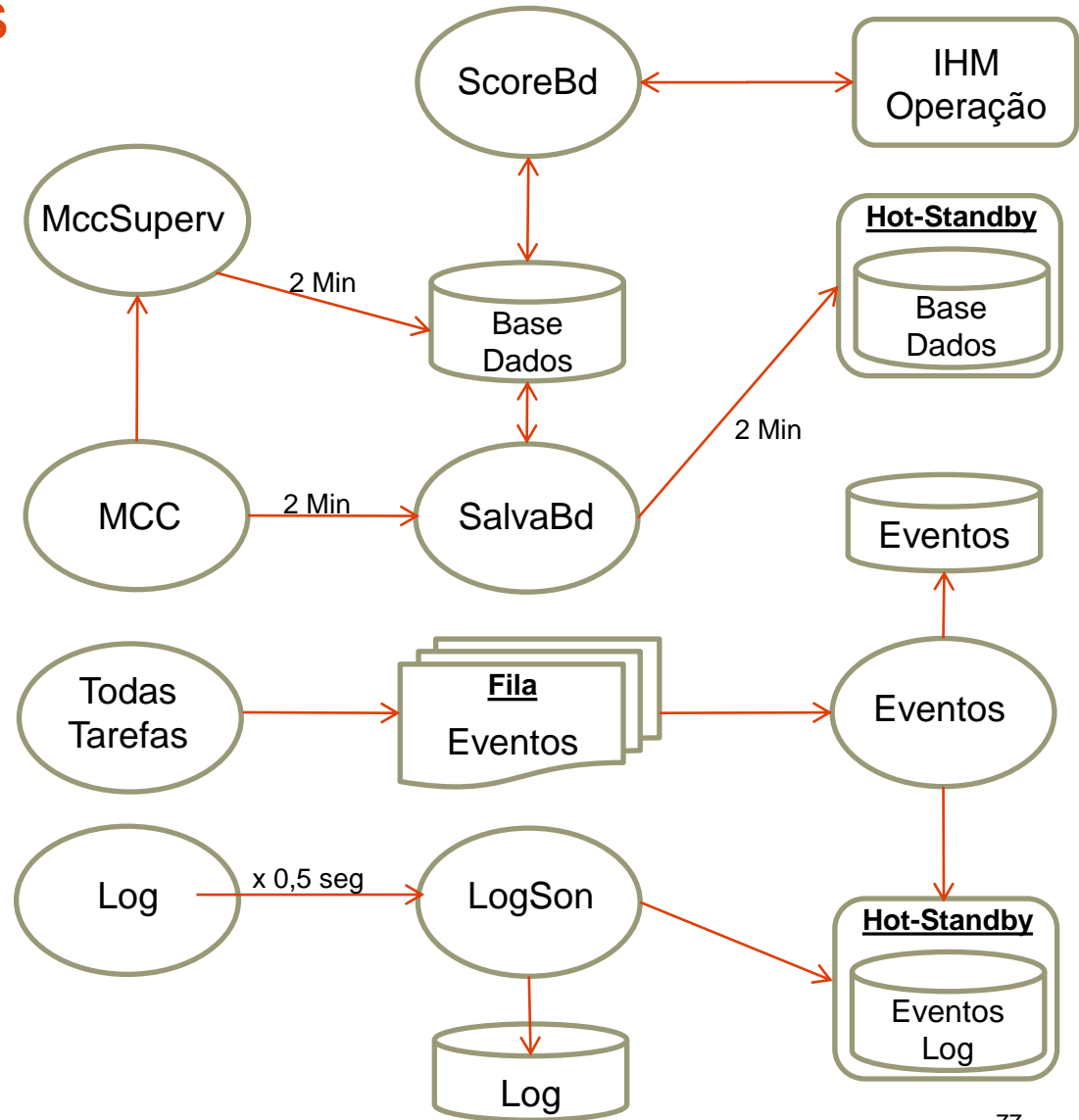
## Ciclos de Execução do Algoritmo

- Ciclo de Leitura (MCP)
  - A cada 0,5 seg através da tarefa MCP
    - Ativada pelo DrvSad
    - Implementação dos cálculos primários
    - Ativa o McpCorrida e mais 6 módulos residentes de ciclo de leitura
- Ciclo de Controle (MCC)
  - A cada 240 ciclos de leitura ou 2 min
    - Ativada pelo MCP
    - Implementa os cálculos de controle de resistência
    - Ativa várias tarefas e mais 6 módulos residentes de ciclo de controle

# Arquitetura do Sistema

## Base de Dados e Filas

- SalvaBd copia base de dados para hot-standby a cada 2 min
- Leitura e gravação via IHM Operação através do ScoreBd:
  - Config. Parâmetros
  - Opções de Operação
- MccSuperv trata dados históricos diários/turno
- Todas tarefas do sistema geram eventos
- Logs gravados a cada 2 min com base de tempo múltipla de 0,5 seg



# Arquitetura do Sistema

## Estruturas de Dados e Diretórios

### Área Comum

#### AVC

#### COMUM

- AVL
- Tabelas Eliminação de Efeito anódico
- Tabelas de Incrementos
- Descritores de Eventos
- Descritores de Logs Ativos
- Cadastro de Tarefas Residentes
- Relação de Bits de I/O
- Flags de Estado das Cubas
- Variáveis de Supervisão de Turno
- Variáveis de Supervisão Diária

### Diretórios

#### • Base de Dados

- AVC
- comum
- arqvar.dat
- MsgStr.CBA
- descr\_conv
- ev\_cuba
- grupo\_cubas
- rede\_qnx
- \*graf\*
- superv\_avc

#### • Eventos

- Eventos\_dd\_mm\_aaaa

#### • Hist

- hist\_dd\_mm\_aaaa
- turno\_t\_dd\_mm\_aaaa

#### • Logrelcuba

- Arquivos de relatorio de alarmes

#### • Log e Log/Aux

- arq\_log.xxx
- arq\_log.xxx.E

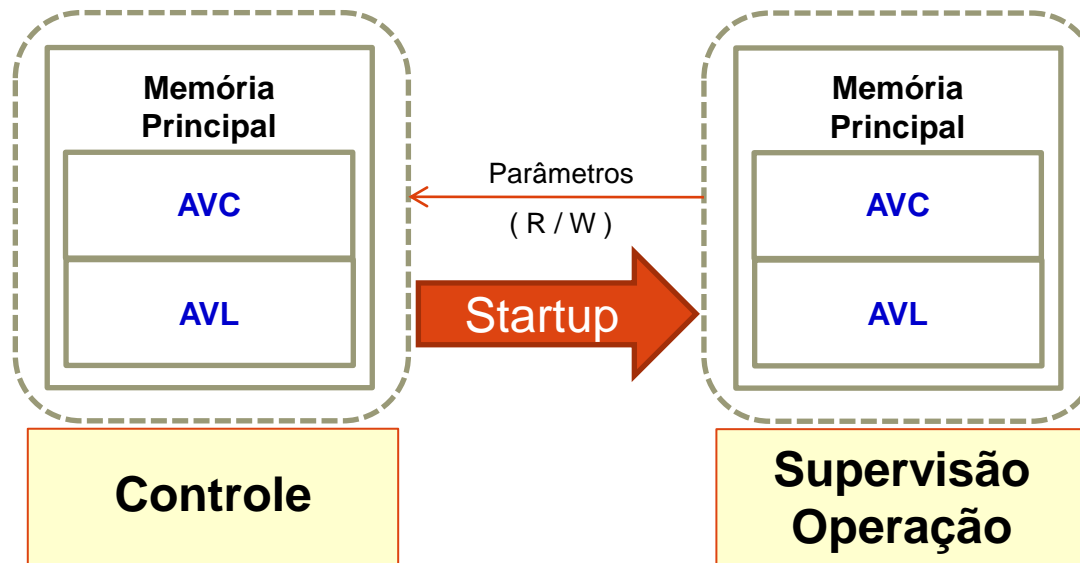
# Arquitetura do Sistema

---

## Laboratório

- Partir o sistema de controle na máquina virtual QNX\_VM-1
- Partir o sistema de controle Hot-Standby na máquina virtual QNX\_VM-2
- Acompanhar a atualização do Hot-Standby após o startup
- Verificar quais arquivos são atualizados em cada ciclo de controle depois que o Hot-Standby ficou pronto para assumir o sistema

## Filosofia – Programas de Configuração



- **Startup:** Base de Dados copiada **Controle** → **Operação/Supervisão**
- Gravação de Parâmetros:
  1. **Base de Dados:** **Controle** → **Operação/Supervisão**
  2. Aloca Semáforo
  3. **Base de Dados:** **Operação/Supervisão** → **Controle**
  4. Libera Semáforo



# Arquitetura do Sistema

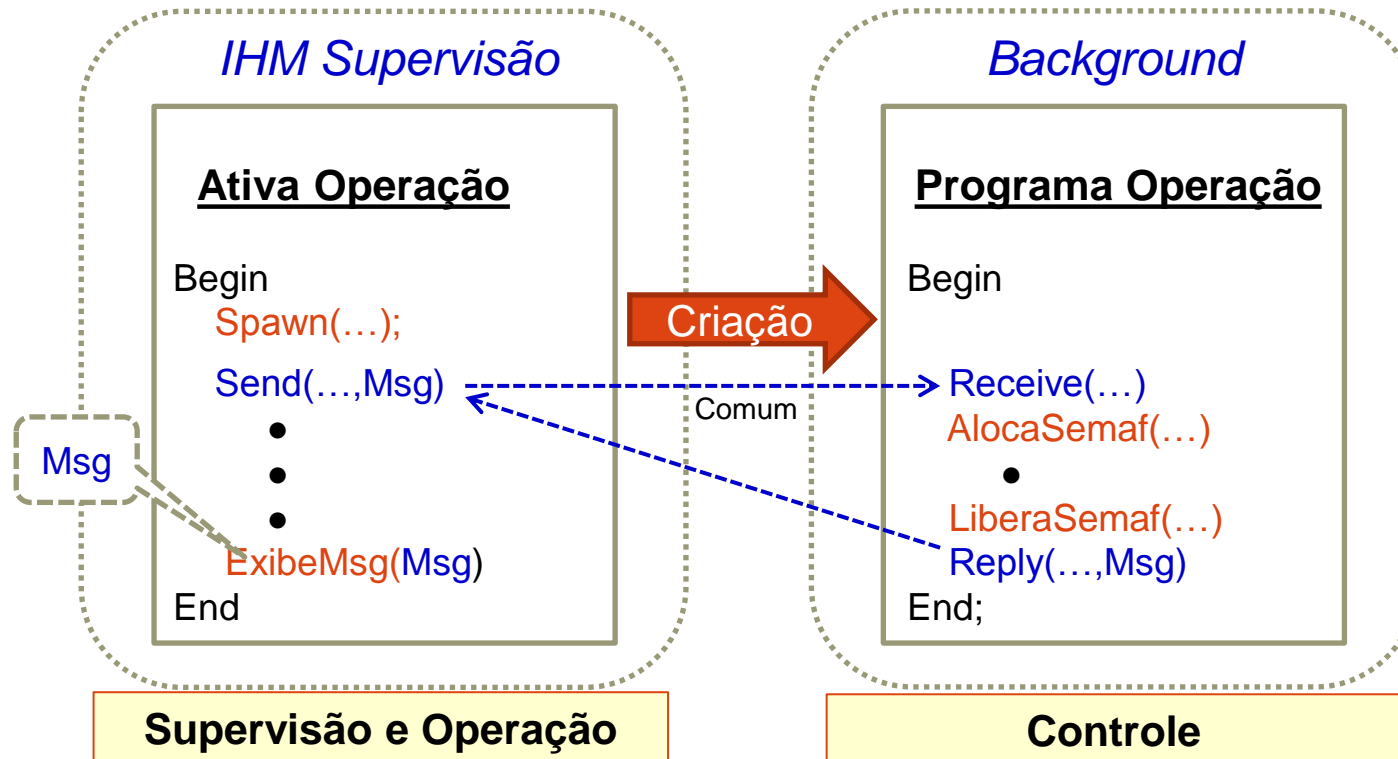
---

## Laboratório

- Carregar o programa de configuração de parâmetros de cuba na IHM de supervisão
- Verificar pelo comando **sin** em qual nodo o programa de configuração está rodando

# Arquitetura do Sistema

## Filosofia – Programas Operação



- IHM Supervisão

- Cria Tarefa via rede
- Envia endereço comum
- Exibe mensagem execução

- Nodo Controle

- Executa programa operação
- Atualiza base de dados
- Retorna mensagem de execução

# Arquitetura do Sistema

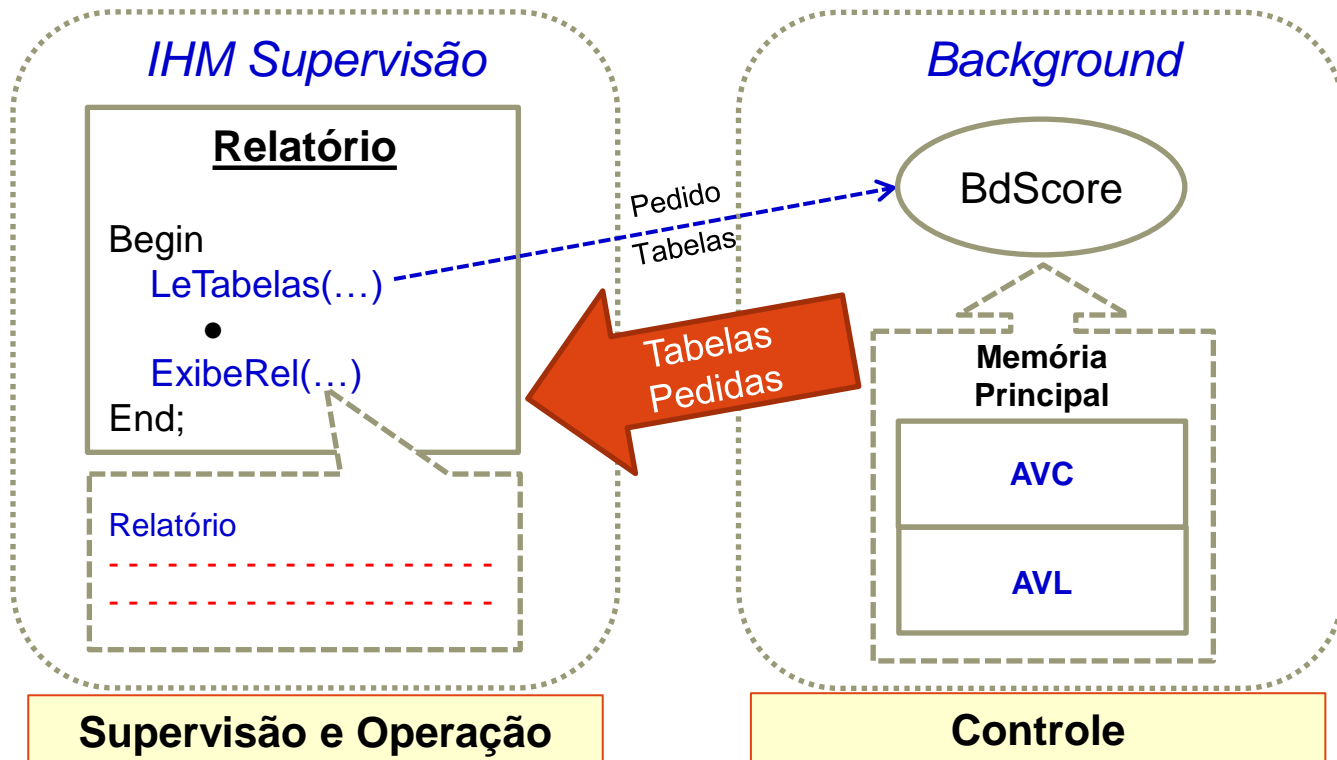
---

## Laboratório

- Colocar um **sleep (40)** no início do programa exemplo de operação do toolkit, recompilar e atualizar no nodo de controle
- Executar o programa de operação
- Verificar pelo comando **sin** em qual nodo o programa de operação está rodando

# Arquitetura do Sistema

## Filosofia – Relatórios (Ciclo de Controle e Históricos)



- **IHM Supervisão**

- Relatório atualiza tabelas
- Processa e Formata saída
- Exibe relatório

- **Nodo Controle**

- Recebe pedido de tabelas
- Atualiza tabelas pedidas

## Laboratório

- Perguntas:
  - Uma tarefa residente do sistema pode atribuir valor a um parâmetro?
  - Preciso desativar o sistema de controle para instalar um programa de operação?
  - Que cuidados preciso tomar na leitura de dados do relatório?
- Executar o relatório de ciclo de controle:
  - Carregar um relatório de ciclo de controle na IHM de supervisão
  - Verificar pelo comando **sin** em qual nodo o relatório está rodando

## Principais Diretórios do Score Runtime

Diretório	Conteúdo
/score/base_dados	Bases de dados do SCORE (avc, comum, etc...)
/score/bin	Comandos do SCORE.
/score/config	Configuração do IHM do SCORE e do ambiente QNX.
/score/erros	Registros de erros ocorridos no SCORE.
/score/eventos	Eventos históricos do SCORE.
/score/exec	Programas executáveis e descritores de relatórios.
/score/help	Textos de ajuda do IHM.
/score/hist	Históricos diários e de turno do SCORE.
/score/log	Logs gerados pelo módulo histórico.
/score/logrelcuba	Logs gerados pelo relatório de alarmes.
/score/telas	Telas da interface gráfica do SCORE
/score/testes	Programas de testes da remota ATN 7.0 e da interface analógica ATN 1.4
/score/tmp	Arquivos gerados temporariamente pelo SCORE
/score/util	Utilitários do SCORE.

# Arquitetura do Sistema

## Principais Diretórios de Fontes

Diretório	Conteúdo
/score/srcs/entdados	Bibliotecas de edição
score/srcs/eventos	Prólogos de eventos
/score/srcs/ihm	Bibliotecas de ihm de console
/score/srcs/ihm_windows	Bibliotecas de ihm windows
/score/srcs/include	Prólogos dos programas.
/score/srcs/interfrem	Biblioteca de interface com a remota
/score/srcs/lib_ihm_ph	Biblioteca da ihm Photon
/score/srcs/lib_ph	Biblioteca da ihm Photon
/score/srcs/lib_score	Documentação da biblioteca de funções genéricas.
/score/srcs/mcc/MccCba	Algoritmo de ciclo de controle da CBA
/score/srcs/mcp/McpCbaTk	Algoritmo de ciclo de leitura da CBA
/score/srcs/ope	Exemplo de programas de operação.
/score/srcs/queue	Biblioteca, prólogo e detalhamento das funções do queue
/score/srcs/rel	Bibliotecas e exemplos de relatórios
/score/srcs/util	Programas fontes de utilitários do sistema.

## Principais Prólogos do Sistema

Prólogo	Conteúdo
<b>BdScore.h</b>	Definições de acesso à base de dados do sistema.
<b>Definicao.h</b>	Operadores da linguagem C usados no sistema
<b>DescrArqVar.h</b>	Configuração das variáveis do SCORE para acesso pelo módulo histórico, módulo de configuração e opções de exportação.
<b>Eventos.h</b>	Códigos de eventos do sistema.
<b>EventosCba.h</b>	Códigos de eventos específicos da CBA.
<b>EventosMsg.h</b>	Definições de tipos do descritor de eventos.
<b>IniArqVar.h</b>	Definições das estruturas e tipos utilizados pelo DescrArqVar.h.
<b>Listas.h</b>	Protótipos das funções da biblioteca Listas.lib
<b>Macros.h</b>	Definições das macros utilizadas no sistema.
<b>OperacaoBibW.h</b>	Protótipos das funções da biblioteca ConsoleW.lib
<b>Prototipos.h</b>	Protótipos das funções da biblioteca Score.lib
<b>Relatórios.h</b>	Definições de estruturas de dados e constantes dos relatórios.
<b>RelPrototipos.h</b>	Protótipos das funções da biblioteca BibRel.lib
<b>Score.h</b>	Principais estruturas de dados dos arquivos avc e comum.
<b>ScoreCba.h</b>	Estruturas de dados do AVC e AVL específicos do usuário.
<b>ScoreConst.h</b>	Principais definições de constantes do sistema.
<b>ScoreErros.h</b>	Códigos de erros e suas respectivas mensagens.
<b>ScoreMsgs.h</b>	Definições da base de dados de strings para estrutura multi-língua do sistema.
<b>TiposOpeW.h</b>	Definições das estruturas de dados e constantes da IHM Windows.
<b>VarsComum.h</b>	Definições dos ponteiros das estruturas de dados do SCORE.



# Arquitetura do Sistema

---

## Laboratório

- Percorrer os diretórios de mcp, mcc, operação e relatórios
- Entrar no diretório include e editar os prólogos com os seguintes conteúdos
  - Declaração da área comum
  - Códigos de eventos
  - Macros de acesso ao AVC, AVL e variáveis históricas diárias e de turno
  - Protótipo das funções da biblioteca LibScore
- Abrir o arquivo **.doc** do diretório **lib\_score** e verificar seu conteúdo
- Fazer um programa que exhibe na tela o tamanho dos principais tipos de variáveis utilizados pelo sistema (byte, short, int, long, float, double)

Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Estrutura de Dados

---

## Variáveis do Sistema

- Variáveis de Processo
  - Informações de processo:
    - **AVL** : Arquivo de Variáveis de Linha
    - **AVC** : Arquivo de Variáveis de Cuba
  - Classes de variáveis:
    - Parâmetros de configuração
    - Variáveis de trabalho
    - Flags de estado
    - Variáveis de supervisão diária/turno
- Comum
  - Descritor de Eventos
  - Tabelas de incrementos
  - Tabelas de supressão de efeito anódico
  - Lista de tarefas residentes
  - Relação de bits de I/O
  - Descritor de Logs Ativos

## Passos de Criação de Variáveis no AVL e AVC

- Criação, Utilização e Instalação
  1. Declarar a variável na respectiva estrutura do prólogo **ScoreCba.h**
  2. Incluir a variável no arquivo **DescrArqVar.h** para ficar acessível pela IHM de supervisão do sistema
  3. Incluir o texto de ajuda da variável no arquivo de help no respectivo arquivo de configuração de linha ou cuba.
  4. Incluir a referência e atribuição da variável nos programas
  5. Recompilar todo o sistema Score e testar a nova versão da aplicação
  6. Preparar um upgrade do *Runtime* do sistema com os arquivos alterados, incluindo o arquivo **ArqVar\*.dat** e instalar

# Estrutura de Dados

## Exemplo de Criação de Parâmetro AVC

- Diretrizes da declaração
  - Declaração na respectiva estrutura do prólogo **ScoreCba.h**
  - O tamanho da estrutura é fixo, por isso a posição das variáveis não pode ser deslocada.
  - **Exemplo:** criar variável **VIncMax** tipo **float** (4 bytes) na 1ª tabela de parâmetros de cuba **t\_par1\_user**
- **1º Passo:** declarar a variável na estrutura **t\_par1\_user**
  - Estrutura antes da inclusão da variável:

```
volatile struct par1_user_avc /* 1a. tabela de parametros do usuario */
{
    /* Parametros de Usuario */
    float Bemf; /* Força contra eletromotriz */
    float VSetPoint; /* Tensao set-point */
    float Livre1; /* Variavel disponivel */
    float EaVoltLim; /* Tensão limite de EA */
    char ResParUser[496]; /* reserva */
} t_par1_user;
```

# Estrutura de Dados

## Exemplo de Criação de Parâmetro AVC

- **1ª Opção** : declarar a variável no final da estrutura
  1. Incluir uma linha com a declaração da nova variável
  2. Deduzir o número de bytes da variável na reserva **ResParUser**
- Estrutura depois da inclusão da variável:

```
volatile struct par1_user_avc /* 1a. tabela de parametros do usuario */
{
    /* Parametros de Usuario */
    float          Bemf;          /* Força contra eletromotriz */
    float          VSetPoint;     /* Tensao set-point */
    float          Livre1;        /* Variavel disponivel */
    float          EaVoltLim;     /* Tensão limite de EA */
    float          VIncMax;       /* Tensão limite de incremento */
    char           ResParUser[492]; /* reserva 496 - 4 = 492 */
} t_par1_user;
```

# Estrutura de Dados

## Exemplo de Criação de Parâmetro AVC

- **2ª Opção** : substituir uma variável que não é mais utilizadas
  1. Incluir uma linha com a declaração da nova variável
- Estrutura depois da inclusão da variável:

```
volatile struct par1_user_avc /* 1a. tabela de parametros do usuario */
{
    /* Parametros de Usuario */
    float          Bemf;          /* Força contra eletromotriz */
    float          VSetPoint;     /* Tensao set-point */
    float          VIncMax;       /* Substituída Livre1 por VIncMax */
    float          EaVoltLim;     /* Tensão limite de EA */
    char           ResParUser[496]; /* reserva */
} t_par1_user;
```

# Estrutura de Dados

## Exemplo de Criação de Parâmetro AVC

- **2º Passo**: incluir a variável no arquivo *DescrArqVar.h*
- Cada linha refere a uma variável configurada pelos parâmetros do quadro

Campo	Tipo	Descrição
NomeVar	ascii	Nome do parâmetro
UeVar	ascii	Unidade de engenharia
EndrVar	ponteiro	Endereço do parâmetro
NivelSenha	define	Nível de senha do parâmetro
TipoVar	define	Tipo do parâmetro
FmtVar	ascii	Formato do parâmetro
LimInf	real	Limite Inferior
LimSup	real	Limite Superior
FatConv	real	Fator de conversão para unidade desejada
Classe	define	Classe do parâmetro
FlagAlg	define	Algoritmo ao qual o parâmetro pertence
Rotulo	ascii	Rótulo do parâmetro no relatório de alarmes
Tag	ascii	Tag do parâmetro referido no relatório de alarmes
GrupoVar	define	Grupo do parâmetro

- O domínio de valores está declarado no arquivo *IniArqVar.h*



## Exemplo de Criação de Parâmetro AVC

- A variável no **DescrArqVar.h** deve ser declarada no grupo correspondente à estrutura do **ScoreCba.h** na qual foi declarada, conforme quadro ao lado.

Assim, a variável **VIncMax** declarada em **t\_par1\_user** no **ScoreCba.h** será configurada em **VarPar1** no **DescrArqVar.h**

DescrArqVar.h	Score.h
VarParamAvc	t_param_avc
VarSuperv	t_avc_supervisao
VarEst	t_est_avc
VarAvl	t_avl
VarPar1	t_par1_user
VarPar2	t_par2_user
VarUser1	t_user1_avc
VarUser2	t_user2_avc
VarUser3	t_user3_avc
VarUser4	t_user4_avc

- **3º Passo:** incluir o texto de ajuda no arquivo **ParCubasW.hlp**
- O nome da variável entre [ ] funciona como uma chave de busca para acessar a descrição
  - O tamanho do texto de help é fixo e limitado a 5 linhas.

## Exemplo de Criação de Parâmetro AVC

➤ **4º Passo**: incluir a referência da variável nos programas

- Declaração da estrutura de parâmetros:

```
volatile struct arq_var_cubas /* Arquivo de variaveis das cubas */
{
    •
    t_par1_user          ParUser1[MAX_CUBAS];
    t_par2_user          ParUser2[MAX_CUBAS];
    •
} t_avc;
```

- Parâmetro declarado em *t\_par1\_user*:

```
If (AVC.User1(Cuba).CCicCont.Vinc > AVC.Par1User[Cuba].VIncMax)
    AVC.User1(Cuba).CCicCont.Vinc = AVC.Par1User[Cuba].VIncMax;
```

- Parâmetro declarado em *t\_par2\_user*:

```
If (AVC.User1(Cuba).CCicCont.Vinc > AVC.Par2User[Cuba].VIncMax)
    AVC.User1(Cuba).CCicCont.Vinc = AVC.Par2User[Cuba].VIncMax;
```

# Estrutura de Dados

---

## Exemplo de Criação de Parâmetro AVC

- **5º Passo:** recompilar todo o sistema Score e testar a nova versão da aplicação na plataforma de desenvolvimento  
# **GeraScore**
- **6º Passo:** preparar e instalar o upgrade de *Runtime* incluindo dentre os arquivos alterados o **ArqVar.dat** e **ArqVar.dat** do diretório **/score/base\_dados**.
  - A instalação dos arquivos arqvar.dat deve ser feita utilizando setup devido à diferença entre a ATN7 e ATN8.
- **Observação:** esse exemplo vale para qualquer estrutura de dados da estrutura AVC e AVL do prólogo **ScoreCba.h**, desde que aplicado na respectiva estrutura de dados.

## Roteiro de Criação de Variáveis de Linha (AVL)

- Criação, Utilização e Instalação (Exemplo AVC)
  1. Declaração da variável na estrutura adequada do prólogo **ScoreCba.h** em **t\_avl** :
    - **t\_par\_avl\_usu** ▶ **Par** : parâmetro de configuração
    - **t\_trab\_avl\_usu** ▶ **Trab** : variável de trabalho
    - **t\_sup\_avl\_usu** ▶ **Sup** : variável de supervisão diária/turno
  2. Incluir a variável no arquivo **DescrArqVar.h** para ficar acessível pela opção de **Configuração de Parâmetros de Linha** da IHM (se necessário).
  3. Incluir texto de ajuda da variável no arquivo **ParLinhaW.hlp** (se necessário)
  4. Incluir a referência da variável **NomeVar** nos programas
    - **AVL.Par.NomeVar** : parâmetro de configuração
    - **AVL.Trab.NomeVar** : variável de trabalho
    - **AVL.Sup.NomeVar** : variável de supervisão diária
    - **AVL\_TURNO.Sup.NomeVar** : variável de supervisão de turno
  5. Recompilar todo o sistema Score e testar a nova versão da aplicação
  6. Gerar e instalar a nova versão do sistema e incluindo arquivo **IniArqVar.dat**

## Roteiro de Criação de Variáveis de Cubas (AVC)

- Criação, Utilização e Instalação (Exemplo AVC)
  1. Declaração da variável na estrutura adequada do prólogo **ScoreCba.h** em **t\_avc** :
    - **t\_par1\_user** ▶ **Par1User[MAX\_CUBAS]** : 1ª tabela de parâmetros
    - **t\_par2\_user** ▶ **Par2User[MAX\_CUBAS]** : 2ª tabela de parâmetros
    - **t\_user1\_avc** ▶ **User1[MAX\_CUBAS]** : 1ª tabela de variáveis de trabalho
    - **t\_user2\_avc** ▶ **User2[MAX\_CUBAS]** : 2ª tabela de variáveis de trabalho
    - **t\_user3\_avc** ▶ **User3[MAX\_CUBAS]** : 3ª tabela de variáveis de trabalho
    - **t\_user4\_avc** ▶ **User4[MAX\_CUBAS]** : 4ª tabela de variáveis de trabalho
    - **t\_est\_usu** ▶ **Est** : variável de status
    - **t\_sup\_avc\_usu** ▶ **Sup** : variável de supervisão diária/turno

## Roteiro de Criação de Variáveis de Cubas (AVC)

- Criação, Utilização e Instalação (Exemplo AVC)
  2. Incluir a variável no arquivo **DescrArqVar.h**, possibilitando o acesso através da opção de **Configuração de Parâmetros de Cuba** da IHM.
  3. Incluir o texto de ajuda do parâmetro (somente) no arquivo **ParCubasW.hlp**

## Roteiro de Criação de Variáveis de Cubas (AVC)

- Criação, Utilização e Instalação (Exemplo AVC)
  - 4. Incluir a referência da variável **NomeVar** nos programas
    - **AVC.Par1User[Cuba].NomeVar** : 1ª tabela de parâmetros
    - **AVC.Par2User[Cuba].NomeVar** : 2ª tabela de parâmetros
    - **AVC.User1[Cuba].NomeVar** : 1ª tabela de variáveis de trabalho
    - **AVC.User2[Cuba].NomeVar** : 2ª tabela de variáveis de trabalho
    - **AVC.User3[Cuba].NomeVar** : 3ª tabela de variáveis de trabalho
    - **AVC.User4[Cuba].NomeVar** : 4ª tabela de variáveis de trabalho
    - **EST\_AVC(Cuba).Est.NomeVar** : variável de status
    - **SUPERV\_AVC(Cuba).Sup.NomeVar** : variável de status
    - **AVL.Sup.NomeVar** : variável de supervisão diária
    - **SUPERV\_AVC\_TURNO(Cuba).Sup.NomeVar** : variável de supervisão de turno

## Roteiro de Criação de Variáveis de Cubas (AVC)

- Criação, Utilização e Instalação (Exemplo AVC)
  5. Recompilar todo o sistema Score e testar a nova versão da aplicação
  6. Preparar um upgrade do *Runtime* do sistema com os arquivos alterados, incluindo o arquivo **ArqVar\*.dat** e instalar



# Estrutura de Dados

---

## Relação de Estruturas de Variáveis

- Variáveis de Linha (AVL) em **t\_avl**
  - **t\_par\_avl\_usu** ▶ estrutura de parâmetros de linha
  - **t\_trab\_avl\_usu** ▶ estrutura de variáveis de trabalho de linha
  - **t\_sup\_avl\_usu** ▶ estrutura de variáveis diárias/turno de linha
- Variáveis de Cuba (AVC) em **t\_avc**
  - **t\_par1\_user** ▶ 1ª estrutura de parâmetros de cuba
  - **t\_par2\_user** ▶ 2ª estrutura de parâmetros de cuba
  - **t\_user1\_avc** ▶ 1ª estrutura de variáveis de trabalho de cuba
  - **t\_user2\_avc** ▶ 2ª estrutura de variáveis de trabalho de cuba
  - **t\_user3\_avc** ▶ 3ª estrutura de variáveis de trabalho de cuba
  - **t\_user4\_avc** ▶ 4ª estrutura de variáveis de trabalho de cuba
  - **t\_sup\_avc\_usu** ▶ estrutura de supervisão diária/turno de cuba
  - **t\_est\_usu** ▶ estrutura de variáveis de status de cuba

## Laboratório

- Criar as seguintes variáveis na base de dados:
  - **ParLTeste:** parâmetro de linha (int)
  - **VarCTeste:** variável de cuba (float)
  - **FlagCTeste :** flag de status (byte)
- Exibir as variáveis criadas de acordo com os seguintes critérios:
  - **ParLTeste:** somente no programa de configuração de linha, não pode entrar no relatório de parâmetros
  - **VarCTeste:** programa de configuração e relatório de parâmetros de cuba
  - **FlagCTeste :** relatório de parâmetros de cuba e na área de status do relatório de alarmes de cuba

Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

**Algoritmo de Controle**

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Algoritmo de Controle

---

## Introdução

- Ciclos de execução
  - **Ciclo de Leitura:** MCP – Módulo de Cálculos Primários
  - **Ciclo de Controle:** MCC – Módulo de Ciclo de Controle
- Implementação dos algoritmos
  - Arquitetura das tarefas (MCP e MCC)
  - Implementação das tarefas
  - Compilação e depuração da tarefa
  - Configuração do startup da tarefa na base de dados
- Criação de eventos
  - Declaração do código do evento no prólogo
  - Configuração dos parâmetros na base de dados
  - Geração do evento no programa

# Algoritmo de Controle

## MCP e MCC - Implementação da Tarefa

- Esboço do Algoritmo
  - Declaração de variáveis e estruturas de dados
  - Algoritmo da tarefa (lógicas de programação, comunicação entre tarefas, funções, etc...)
- Declaração de Variáveis

### Início

```
const      INICIALIZACAO=1
const      CICLO_CONTROLE=2
const      CICLO_LEITURA=3
const      DESABILITOU_LINHA=4
tipo MsgSinc = registro
                inteiro      : Origem;
                inteiro      : Acao;
                tipo v        : arranjo [1:8] inteiro;
                fim registro

v          : Parametro;
MsgSinc    : Msg;
```

# Algoritmo de Controle

## MCP - Implementação da Tarefa

- Algoritmo do MCP

- Receber mensagem com endereço da área comum (SCORE).
- Inicializar ponteiros para área comum.
- Fazer consistência dos parâmetros da chamada da tarefa.

enquanto (VERDADEIRO) faça

- Receber mensagem de ativação do DRVSAD (0,5 seg)
- Pegar hora atual

escolha (Msg.Acao)

caso

INICIALIZACAO :

- Executar procedimentos de inicialização.
- Aguardar partida do sistema se *Hot-Standby*.

caso

CICLO\_LEITURA :

- ProcessaCicloLeitura(CubaInicial, CubaFinal)

caso

DESABILITOU\_LINHA :

- Tratar desabilitação da linha.

fim escolha

- Enviar resposta a mensagem de ativação

fim enquanto

Fim

# Algoritmo de Controle

---

## MCP - Implementação da Tarefa

- Cuidados na Implementação
  - O MCP executa um ciclo de leitura a cada 0,5 segundo, portanto:
    - Qualquer **comando que aguarde confirmação** ou coloque a **tarefa em espera desativa o sistema**
    - A **tempo de execução** da tarefa sempre deve ser **inferior a 0,5 seg**, caso contrário o sistema será desativado.
- Ambiente de Desenvolvimento:
  - **Diretório MCP:** [/score/srcs/mcp](#)
  - **Exemplos:** [McpUser.h](#) e [McpUser.c](#)
  - **Makefile:** [makeuser](#)
- O diretório [/score/srcs/mcp/McpCbaTk](#) contém o algoritmo de ciclo de leitura que está em produção na CBA

# Arquitetura do Sistema

---

## Laboratório

- Abrir o diretório [/score/srcs/mcp](#) e analisar seus principais arquivos



# Algoritmo de Controle

## MCC - Implementação da Tarefa

- Algoritmo do MCC

- Receber mensagem com endereço da área comum (SCORE).
- Inicializar ponteiros para área comum.
- Fazer consistência dos parâmetros da chamada da tarefa.

enquanto (VERDADEIRO) faça

- Receber mensagem de ativação MCP (2 min)
- Pegar hora atual

se (Msg.Origem = MCC)

escolha (Msg.Acao)

caso

INICIALIZACAO :

- Executar procedimentos de inicialização.
- Aguardar partida do sistema se *Hot-Standby*.

caso

CICLO\_CONTROLE :

- ProcessaCicloControle(CubaInicial, CubaFinal)

caso

DESABILITOU\_LINHA :

- Tratar desabilitação da linha.

fim escolha

fim se

- Enviar resposta a mensagem de ativação

fim enquanto

Fim

# Algoritmo de Controle

---

## MCC - Implementação da Tarefa

- Cuidados na Implementação
  - O MCC executa um ciclo de controle a cada 2 minutos, portanto:
    - Qualquer **comando que aguarde confirmação** ou coloque a **tarefa em espera desativa o sistema** após 2 minutos
    - A **tempo de execução** da tarefa sempre deve ser **inferior a 2 minutos**, caso contrário o sistema será desativado.
- Ambiente de Desenvolvimento:
  - **Diretório MCP:** [/score/srcs/mcc/MccCba](#)
  - **Exemplos:** [MccUser.h](#) e [MccUser.c](#)
  - **Makefile:** [makeuser](#)
- O diretório [/score/srcs/mcc/MccCba](#) contém o algoritmo de ciclo de controle que está em produção na CBA

# Arquitetura do Sistema

---

## Laboratório

- Abrir o diretório [/score/srcs/mcc](#) e analisar seus principais arquivos

## Implementação, Compilação e Depuração

- Detalhes de implementação
  - **Editor de textos:** **vedit**
  - Linguagem e Implementação
    - Programa em linguagem C, respeitando sua sintaxe e padrões
    - Toda tarefa **.c** tem respectivo prólogo **.h**, ex: **MccUser.c** e **MccUser.h**
  - Compilação:
    - **Compilador:** Watcom C/C++ versão 10.6
    - Relações de dependências declaradas no **makefile**
  - Depuração:
    - **Comando:** **wd**
    - **Diretiva de Compilação:** **-g** (macro DEBUG)
  - Instalação:
    - Instalação de nova tarefa residente na base de dados via **InstalaCtrl** visto no capítulo **Ferramentas de Desenvolvimento**

## Procedimento de Criação de uma Tarefa Residente

- No exemplo seguinte utilizaremos como exemplo uma tarefa de ciclo de leitura **McpUser**, mas lembramos que o procedimento é o mesmo tanto para tarefas de ciclo de leitura e controle.
- Destacamos que esse procedimento deve ser realizado na plataforma de desenvolvimento.
- **1º Passo** : implementar os arquivos fonte **McpUser.h** e **McpUser.c**
- **2º Passo** : criar variáveis no **ScoreCba.h** e configurar no **DescrArqVar.h**
- **3º Passo** : incluir as dependências no **makefile** e compilar:  
    # cd /score/srcs/Mcp/McpCbaTk  
    # GeraScore
- **4º Passo** : instalar a tarefa na base de dados de simulação pelo **InstalaCtrl** visto no capítulo *Ferramentas de Desenvolvimento*

# Criação de Relatórios

---

## Procedimento de Criação de uma Tarefa Residente

- **5º Passo** : depurar e testar a tarefa
- **6º Passo** : preparar um upgrade com os arquivos alterados conforme capítulo *Preparação e Instalação de Upgrades*
  - O procedimento de instalação da partida da tarefa via **InstalaCtrl** deve ser repetido na instalação do 1º micro de controle da redução com o sistema desativado.

# Algoritmo de Controle

---

## Depuração da Tarefa

- Utilitário dumper gera um arquivo de erro **.dmp** no diretório **/score/erros** quando um programa morre por erro de execução
- Usando como exemplo o programa **McpUser**, o arquivo é acessado pelo comando **wd** conforme a sintaxe abaixo:
  - # **Wd -trap=pmd /score/erros/McpUser.dmp**
    - Depurador entra direto no ponto que o programa morreu
    - **Code → Calls:** mostra, em alguns casos, a seqüência de funções chamadas antes do erro.
    - Pode retornar às condições da tarefa antes do erro falha para identificar seu motivo, mostrando por exemplo os valores de variáveis.
- Requisitos da depuração utilizando **.dmp**
  - # **Dumper &**
  - # Programa compilado com diretiva **-g** (macro **DEBUG**)

# Algoritmo de Controle

---

## Laboratório

- Editar a tarefa Mcp e incluir uma divisão por zero
- Compilar e colocar em execução
- Abrir o arquivo **dmp** pelo depurador e verificar o status das variáveis antes da tarefa morrer
- Voltar o sistema à condição normal



# Algoritmo de Controle

---

## Pontos de Atenção

- Do mesmo modo que no micro de controle, o Hot-Standby também carrega todas as tarefas residentes em memória na sua partida. Portanto, quando uma tarefa residente modificada é instalada no Hot-Standby, é necessário sua desativação e reativação carregar a nova versão.
- Qualquer modificação feita no sistema deve ser sempre testada no micro de simulação.
- Se a modificação for muito grande e crítica, é seguro instalá-la primeiro no Hot-Standby, desativando-o e ativando-o novamente. Depois que já está pronto para assumir o controle, transfere-se o controle para ele e mantém o micro que estava como principal desativado na mesma condição que ele ficou para preservar sua base de dados. Somente depois que está garantido que a modificação não vai dar problema é que o outro micro é atualizado e entra como Hot-Standby.

# Algoritmo de Controle

## Criação de Opção de Algoritmo

- **1º Passo** : as opções de algoritmo são criadas na estrutura ***t\_NomeAlgCtr*** do arquivo ***DescrArqVar.h***, definido entre aspas com tamanho máximo de 8 caracteres, como foi criado ***AlgUser1*** no exemplo abaixo:

```
/* algoritmos do usuario */
t_NomeAlgCtr NAlgCtr[] =
{
    /* 09 */      "AlgUser1"
    /* 10 */      " "
    /* 11 */      " "
    /* 12 */      " "
    /* 13 */      " "
};
```

- **2º Passo** : Recompilar o arquivo ***/score/util/IniArqvar*** e instalar os arquivos ***Arqvar.dat*** e ***Arqvar1.dat*** na base de dados do sistema, exibindo o novo algoritmo na tela de pré-seleção.
- **3º Passo** : incluir a referência para o algoritmo nas tarefas.

## Criação de Eventos do Sistema

- Características dos Eventos
  - Registro de ocorrências importantes do sistema
  - Componentes do evento:
    - Cuba ou Faixa de Cubas
    - Mnemônico
    - Mensagem
    - 5 Parâmetros variáveis
- Resumo da Criação de Eventos
  1. Declaração do código do evento no prólogo
  2. Configuração dos parâmetros do evento no descritor
  3. Atualização da configuração do descritor no comum
  4. Inclusão das informações de eventos no arquivo de *help*
  5. Inclusão da função de geração do evento no programa

# Algoritmo de Controle

---

## Criação de Eventos do Sistema

- **1º Passo** : Declarar o Código do Evento no Prólogo
  - Os códigos de eventos são declarados no arquivo **EventosCBA.h** do diretório **/score/srcs/include**
  - Sintaxe da declaração e parâmetros

```
# Define COD_EV           NumEv           /* DescrEv */
```

- Descrição dos parâmetros:
  - **COD\_EV**: nome da constante do evento que será utilizada como referência no código do programa
  - **NumEv**: valor numérico de 1 a 200, onde os valores de 1 a 99 são reservados para o Score e as novas aplicações podem utilizar a faixa de 100 a 200
  - **DescrEv**: descrição sucinta do evento criado

# Algoritmo de Controle

---

## Criação de Eventos do Sistema

- **2º Passo** : configurar os parâmetros do evento no descritor ***DescrEvph*** visto no capítulo ***Ferramentas de Configuração***
- **3º Passo** : Atualizar a configuração do descritor de eventos no arquivo comum da base de dados através do utilitário ***IniDescrEv*** visto no capítulo ***Ferramentas de Configuração***
- **4º Passo** : Incluir as Informações do Evento no Help:
  - Incluir o *mneumônico*, descrição e parâmetros do evento no arquivo de ***HelpEv.hlp*** do diretório ***/score/help***

## Criação de Eventos do Sistema

- **6º Passo** : Incluir a Função de Geração do Evento nos Programas
  - O evento é gerado nos programas através da função **GeraEvento** da biblioteca **LibScore.lib**, conforme sintaxe no quadro abaixo:

Função	
<b>GeraEvento(CodEv, CubaInicial, CubaFinal, Par1, Par2, Par3, Par4, Par5)</b>	
Parâmetros de Chamada	
Parâmetros	Descrição
<b>CodEv</b>	Código do evento (define) declarado no prólogo EventosCBA.h
<b>CubaInicial</b>	Índice da cuba inicial na base de dados do SCORE. Colocar valor -1 quando não existir (evento de linha).
<b>CubaFinal</b>	Índice da cuba final na base de dados do SCORE. Colocar valor -1 quando não existir.
<b>Par1 .. Par5</b>	1º ao 5º parâmetro do evento, tipo variável de acordo com o evento

# Algoritmo de Controle

---

## Pontos de Atenção

- Os parâmetros passados para a função **GeraEvento** devem ser do mesmo tipo definido através do descritor de eventos. Erros nessa passagem de parâmetros podem provocar exceção no módulo e consequentemente a desativação do sistema.
- **Exemplo :**
  - Caso um parâmetro definido como **string** seja passado uma variável tipo **int** ou **float**, o valor da variável será utilizado como ponteiro para **string** com grande probabilidade de gerar exceção de memória.

# Algoritmo de Controle

---

## Laboratório

- Criar o evento **EvTeste** e com a mesma máscara do gerado pelo programa de operação de exemplo
- Substituir o evento no programa de operação
- Preparar o help do evento
- Instalar o evento e verificar se aparece na lista de filtros do relatório de eventos
- Executar o programa de operação e verificar
  - Se o evento foi gerado
  - Se o help do evento apareceu no relatório



Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Criação de Opção de Operação

## Características do Programa de Operação

- Arquitetura padrão :
  - Criados pelo programa **AtivaOperaçãoPH** pela IHM de supervisão
  - Rodam em background nos micros de controle

### Algoritmo do Programa

- Receber mensagem de inicialização do **AtivaOperacao**.
- Pegar endereço da área comum.
- Fazer consistência dos parâmetros de chamada (IHM)
  - se (Parâmetro de Chamada Ok)
    - então
      - para CubaInicial até CubaFinal faça
        - Executar procedimentos específicos da opção de operação.
        - Gerar evento específico da operação executada.
        - Gerar evento de log da operação executada.
    - fim para
  - fim se
- Retornar mensagem para o **AtivaOperacao** com status de execução

# Criação de Opção de Operação

---

## Procedimento de Criação da Opção de Operação

- **1º Passo** : implementar o programa com as funcionalidades do algoritmo
  - Padrão e sintaxe de programação da linguagem C
  - Editor de textos: **vedit**
  - Exemplo de programa de operação : **/score/srcs/ope/cba/OpeExemplo.c**
- **2º Passo** : Incluir as dependências de compilação/linkedição do programa no **makefile** e compilar
  - # cd /score/srcs/ope/cba
  - # GeraScore

# Criação de Opção de Operação

## Procedimento de Criação da Opção de Operação

- **3º Passo** : configurar a chamada do programa na IHM pelo programa ***ConflHMph*** visto no capítulo ***Ferramentas de Configuração*** considerando as seguintes premissas:
  - Campo **Path + Task Name** deve ter o path **/score/exec/AtivaOperacaoPH**
  - O 1º argumento na configuração da IHM (Pag Arg: 1) deve ter a seguinte configuração :  
**Task Arg:** TEXT STRING FIXED HIDDEN  
**Default Value:** Nome do programa de operação **sem path**
- **4º Passo** : depurar e testar o programa
- **5º Passo** : instalar o programa de operação:
  - O programa de operação é carregado toda vez que acionado pela IHM, portanto não precisa desativar o sistema para atualizá-los.
  - Esses programas são instalados no diretório **/score/exec** como os demais

# Criação de Opção de Operação

---

## Depuração de um Programa de Operação

### Preparação do Programa para Depuração

- Incluir no makefile a diretiva **-g** (macro **DEBUG**)
- Preparar o programa para depuração pelo **wd**
  - Incluir o comando **sleep(30)** logo depois de receber a mensagem de sinalização do **AtivaOperacaoPH**
  - Inibir os comandos de alocação e liberação de semáforos
- Compilar e gerar o módulo executável do programa alterado com a diretiva **-g**
- Depurar o programa no ambiente de desenvolvimento com o sistema de controle e interface gráfica rodando

# Criação de Opção de Operação

## Procedimento de Criação da Opção de Operação

### Depuração do Programa em Execução

- Partir o sistema de controle e interface gráfica
- Chamar o programa pela tela de operações da interface gráfica
- Pegar seu *pid* do processo através do comando **sin** imediatamente à ativação. Considerando o nome do processo começado com **Ope**, o comando deve ser executado numa console livre conforme a sintaxe abaixo :  
  
# sin -P Ope
- Pegar o programa com o depurador **wd** passando como parâmetro o *pid* **PidOpe** conforme a sintaxe abaixo :  
  
# wd PidOpe
  - Esse comando **wd** deve ser executado num intervalo de tempo inferior a 30 segundos (**sleep**). Depois de passado esse intervalo o programa é liberado para depuração passo-a-passo dentro do **wd**.

# Criação de Opção de Operação

## Procedimento de Criação da Opção de Operação

### Depuração do Programa Terminado por Erro de Execução

- Utilitário **dumper**
  - Gera um arquivo de depuração do programa **OpeExemplo** de *path* **/score/erros/OpeExemplo.dmp**
  - Deve ter sido ativado anteriormente conforme a sintaxe abaixo :  
**# dumper &**
- Quando programa **OpeExemplo** morre por erro de execução seu arquivo de dump é acessado pelo depurador conforme a sintaxe abaixo  
**# wd -trap=pmd /score/erros/OpeExemplo.dmp**
  - O depurador entra direto no ponto que ocorreu o erro
  - Selecionando as opções **Show → Calls** em alguns casos mostra a sequencia de chamada de funções antes do erro, permitindo voltar à condição anterior e analisar as variáveis.

# Criação de Opção de Operação

---

## Laboratório

- Abrir o arquivo fonte de incrementos de operação
  - Abrir o arquivo fonte
  - Abrir a configuração dessa opção na IHM
- Analisar alguns tópicos da interface de chamada desse programa:
  - Passagem de parâmetros
  - Tratamento de opções de lista
  - Tratamento de strings
  - Tratamento e duração e tempo



Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

Projeto Final

# Criação de Relatórios

---

## Características dos Relatórios

- Relatórios de Ciclo de Controle
  - Arquivos fonte do relatório **RelExemploCC:**
    - **Prólogo** **RelExemploCC.h** : com a declaração das estruturas de dados de cabeçalho, área de dados e rodapé
    - **Programa** **RelExemploCC.c** : com o código fonte do programa
  - Relatório de ciclo de controle:
    - **Código Executável:** **RelExemploCC**
    - **Descritor** **RelExemploCC.dsc** : máscara do relatório
    - **Help** **RelExemploCC.hlp** : arquivo com descrição dos campos do relatório
  - Linkedição com múltiplas bibliotecas gera automaticamente a versão para impressora e tela do relatório

# Criação de Relatórios

---

## Características dos Relatórios

- Relatórios de Supervisão Diários e Turno
  - Arquivos fonte do relatório **RelExemploDia:**
    - **Prólogo** **RelExemploDia.h** : declaração das estruturas de dados de cabeçalho, área de dados e rodapé
    - **Programa** **RelExemploDia.c** : código fonte do programa
    - **Códigos Executáveis:**
      - **RelExemploDia** (diário)
      - **RelExemploTDia** (turno)
    - **Descritores:** máscara do relatório
      - **RelExemploDia.dsc** (diário)
      - **RelExemploTDia.dsc** (turno)
    - **Help** : descrição dos campos do relatório
      - **RelExemploDia.hlp** (diário)
      - **RelExemploTDia.hlp** (turno)
  - Linkedição com múltiplas bibliotecas gera automaticamente a versão de impressora e tela dos relatórios diário e turno

# Criação de Relatórios

## Estruturas de Dados dos Relatórios

- Lay-Out do divido em 3 áreas: *cabeçalho*, *área de dados* e *rodapé*
- Todos relatórios têm um prólogo com extensão “.h” com mesmo nome do fonte “.c”, onde são declaradas as estruturas dados:
  - Declaração do *cabeçalho*:

```
/*-- Cabeçalho --*/
typedef
    struct VarCabec
    {
        char          Selecao[14];
        .
        .
    } t_cabec;
```

# Criação de Relatórios

## Estruturas de Dados dos Relatórios

- Declaração da *área de dados*:

```
/*-- Area de Dados --*/  
typedef  
    struct VarDados  
    {  
        int          NumCuba;  
        float        VIncr;  
        •            •  
        •            •  
    } t_dados;
```

- Declaração do rodapé:

```
/*-- Rodape --*/  
typedef  
    struct VarRodape  
    {  
        float        VIncrMed;  
        •            •  
        •            •  
    } t_rodape;
```

# Criação de Relatórios

---

## Estruturas de Dados dos Relatórios

- Declaração das estruturas no programa fonte “.c”:

```
/*-- Definição de Variáveis Locais --*/
```

```
t_cabec          Cab;  
t_dados          AreaDados[MAX_LINHAS];  
t_rodape         Rodape;
```

# Criação de Relatórios

## Algoritmo dos Relatórios

### Programa Principal

inteiro: NLinhas;  
t cabec : Cab;  
t dados : AreaDados[MAX\_LINHAS];  
t rodape: Rod;

- Inicializar o ambiente gráfico.
- Pegar endereço da área comum.
- Lêr tabelas de dados atualizadas do micro de controle.

se (Leitura de Tabelas Ok)

então

- Fazer consistência dos parâmetros de chamada.

se (Parâmetros de Chamada Ok)

então

se ((NLinhas = GeraDadosRel(Cab,AreaDados,Rod))> 0)

então

- Gerar o arquivo de dados do relatório.
- Exibir as informações do relatório.

fim se

fim se

fim se

- Finalizar ambiente gráfico

# Criação de Relatórios

---

## Algoritmo dos Relatórios

- Relatório de Ciclo de Controle

### **Função** **GeraDadosRel**

inteiro:                NLinhas;  
t cabec :               Cab;  
t dados :               AreaDados[MAX\_LINHAS];  
t rodape:               Rod;

- Montar cabeçalho do relatório.

para CubaInicial ate CubaFinal faça

- Montar área de dados da cuba.

NLinhas = NLinhas + 1;

fim para

- Montar rodapé do relatório.

GeraDadosRel = NLinhas;



# Criação de Relatórios

## Algoritmo dos Relatórios

- Relatório de Supervisão Diária e Turno

### **Função** **GeraDadosRel**

inteiro: NLinhas;  
t cabec : Cab;  
t dados : AreaDados[MAX\_LINHAS];  
t rodape: Rod;

- Montar cabeçalho do relatório

para DataInicial ate DataFinal faça

- Validar a data tratada

se (Data é válida)

então

- Fazer leitura do arquivo de supervisão da data tratada

para CubaInicial ate CubaFinal faça

- Montar área de dados da cuba para o dia tratado

NLinhas = NLinhas + 1;

fim para

fim se

fim para

- Montar rodapé do relatório

# Criação de Relatórios

---

## Procedimento de Criação de um Relatório

- No exemplo seguinte utilizaremos como exemplo o relatório diário **RelExemploDia**, mas lembramos que o procedimento é o mesmo tanto para relatório de ciclo de controle quanto de turno
- **1º Passo** : implementar os arquivos fonte **RelExemploDia.h** e **RelExemploDia.c**
- **2º Passo** : criar variáveis e incluir cálculos no algoritmo de controle se necessário
- **3º Passo** : incluir as dependências no makefile para gerar os relatórios diários e turno e compilar

```
# cd /score/srcs/rel/tela
# GeraScore
```
- **4º Passo** : criar os descritores **RelExemploDia.dsc** (diário), **RelExemploTDia.dsc** (turno), etc ...

# Criação de Relatórios

---

## Procedimento de Criação de um Relatório

- **5º Passo** : criar os arquivos de help **RelExemploDia.hlp** (diário), **RelExemploTDia.hlp** (turno), etc ...
- **6º Passo** : configurar a chamada do programa na IHM pelo **ConfIHMph** visto no capítulo *Ferramentas de Configuração*.
- **7º Passo** : depurar e testar o relatório
- **8º Passo** : preparar um upgrade com os arquivos alterados conforme capítulo *Preparação e Instalação de Upgrades*
  - Os relatórios são carregados em memória somente quando chamados via IHM. Portanto, se não houver tarefa residente incluída no upgrade não precisa desativar o sistema para a instalação.

# Criação de Relatórios

---

## Depuração de Relatórios

- **Preparação do Programa para Depuração**
  - Incluir no makefile a diretiva **-g** (macro **DEBUG**)
  - Preparar o programa para depuração pelo **wd**
    - Incluir o comando **sleep(30)** logo no início do programa
    - Inibir os comandos de alocação e liberação de semáforos
  - Compilar e gerar os módulos executáveis do programa alterado com a diretiva **-g**
  - Depurar o programa no ambiente de desenvolvimento com o sistema de controle e interface gráfica rodando

# Criação de Relatórios

---

## Depuração de Relatórios

### Depuração do Relatório em Execução

- Partir o sistema de controle e interface gráfica
- Chamar o programa pela tela de relatórios da interface gráfica
- Pegar seu *pid* do processo através do comando **sin** imediatamente à ativação. Considerando o nome do processo começado com **Rel**, o comando deve ser executado numa console livre conforme a sintaxe abaixo :  
  
# sin -P Rel
- Pegar o programa com o depurador **wd** passando como parâmetro o *pid* **PidRel** conforme a sintaxe abaixo :  
  
# wd PidRel
  - Esse comando **wd** deve ser executado num intervalo de tempo inferior a 30 segundos (**sleep**). Depois de passado esse intervalo o programa é liberado para depuração passo-a-passo dentro do **wd**.

# Criação de Relatórios

---

## Depuração de Relatórios

### Depuração do Programa Terminado por Erro de Execução

- Utilitário **dumper**
  - Gera um arquivo de depuração do programa **RelUserDia** de *path* **/score/erros/ RelUserDia.dmp**
  - Deve ter sido ativado anteriormente conforme a sintaxe abaixo :  
**# dumper &**
- Quando programa **RelUserDia** morre por erro de execução seu arquivo de dump é acessado pelo depurador conforme a sintaxe abaixo  
**# wd -trap=pmd /score/erros/ RelUserDia.dmp**
  - O depurador entra direto no ponto que ocorreu o erro
  - Selecionando as opções **Code → Calls**, em alguns casos mostra a sequencia de chamada de funções antes do erro, permitindo voltar à condição anterior e analisar as variáveis.

# Criação de Relatórios

---

## Laboratório

- Abrir o arquivo fonte do exemplo de relatório de ciclo de controle e analisar
  - Montagem do cabeçalho, área de dados e rodapé
  - Função de leitura de tabelas
- Abrir o arquivo fonte do exemplo de relatório de supervisão e analisar
  - Varredura da função de montagem dos dados
  - Localizar a macro de acesso às variáveis de supervisão e entender seu detalhamento no prólogo Macros.h
  - Abrir o makefile e identificar como é gerado o relatório diário e turno

Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

Criação de Opção de Operação

Criação de Relatórios

**Preparação e Instalação de Upgrades**

Projeto Final



# Preparação e Instalação de Upgrades

---

## Resumo da Instalação das Alterações

- Procedimento de instalação de um upgrade:
  - Gerar o arquivo de upgrade e copiar para os micros que serão atualizados
  - Fazer backups do sistema como contingência antes de fazer a instalação
  - Instalar o upgrade num micro de controle e supervisão, deixando o outro micro de controle e demais de supervisão desativados como contingência até validar a nova versão.
  - Instalar no outro micro de controle e demais micros de supervisão depois que a nova versão estiver homologada.
  - Atualizar os backups da plataforma de desenvolvimento e runtime do sistema.
  - Atualizar o sistema das demais reduções seguindo o mesmo procedimento da 1ª redução instalada.

# Preparação e Instalação de Upgrades

## Procedimento de Preparação do Upgrade

- **1º Passo** : editar o arquivo `score/altcba/altcba.txt` e incluir versão, data e alterações implementadas conforme o padrão abaixo:

```
ALTERACOES EFETUADAS NO SCORE APARTIR DA VERSAO DE 08/05/98
                VERSAO 4.41

1) Alterado numero da versao de 4.40 para 4.41
2) Correcao da impressao de hardcopy em impressora colorida.
```

- **2º Passo** : preparar a lista de arquivos alterados
  - Gerar a lista pelo comando `filesd` conforme sintaxe abaixo:  
**# filesd AAAAMMDDHHMM dir -v**  
**# filesd 201209250000 /score/ | sort >/tmp/lista**  
No exemplo são listados todos arquivos alterados a partir de 25/09/2012 às 00:00 ordenados pelo path e gravados no arquivo `/tmp/lista`

# Preparação e Instalação de Upgrades

## Procedimento de Preparação do Upgrade

- Limpar da lista os diretórios abaixo deixando somente o necessário:
  - **Base de Dados:** **somente arquivos** *MsgStrCBA\**, *descr\_ev* e *Arqvar\*.dat*. Os arquivos *Avc*, *cnfrelgraf*, *comum*, *evcuba*, *grupocubas*, *relgraf*, *superv\_avc* e *valgrafmini* nunca podem ir no upgrade
  - **Hist, eventos, erros, exporta\_dados, log, log\_rel\_cuba:** nunca vão
- Incluir no arquivo */tmp/lista* os arquivos *install\_msg* (1ª linha), *altscore.txt* e *setup* se necessário (última linha)
- Copiar */tmp/lista* para o path */score/altcba/altrcbavxxxx\_ddmmaaaa*, como por exemplo */score/altcba/altrcbav0015\_25092012*
- **3º Passo** : preparar o arquivo de upgrade
  - Gerar o upgrade a partir da lista com mesmo path com a extensão **.tgz**, ou seja, */score/altcba/altrcbav0015\_25092012.tgz*:
    - # cd /score/altcba
    - # upgrade altrcbavxxxx\_ddmmaaaa altrcbavxxxx\_ddmmaaaa.tgz
    - # upgrade altrcbav0015\_25092012 altrcbav0015\_25092012.tgz

# Preparação e Instalação de Upgrades

## Procedimento de Preparação do Upgrade

- **4º Passo** : copiar a lista e o arquivo de upgrade para o diretório `/score/altcba` do nodo de controle e supervisão que será atualizado
  - **Observação:** um exemplo pode ser visto em `/score/altcba`
- **5º Passo** : fazer os backups de segurança antes de começar a instalação a partir do micro de controle (Hot-Standby) que será instalado:
  - Base de dados
  - Runtime completo
  - Backup de runtime baseado na lista de arquivos que será instalada, caso precise retornar à versão anterior rapidamente:

**# cd /score/altcba**

**# upgrade altcbavxxxx\_ddmmaaaa altcbavxxxx\_ddmmaaaa.ant.tgz**

**# upgrade altcbav0015\_25092012 altcbav0015\_25092012.ant.tgz**

# Preparação e Instalação de Upgrades

---

## Laboratório

- Forçar a compilação do sistema
- Preparar setup de instalação do descritor de eventos e [ArqVar\\*.dat](#)
- Preparar um arquivo de upgrade com os seguintes arquivos:
  - Abertura com o [install\\_msg](#)
  - Arquivos compilados
  - Descritor de eventos e [ArqVar\\*.dat](#)
  - Setup de instalação

# Preparação e Instalação de Upgrades

---

## Instalação do Upgrade no Hot-Standby

- **1º Passo** : desativar o micro Hot-Standby
  - Logo depois da atualização a cada 2 minutos no diretório `/score/base_dados`, o micro Hot-Standby deve ser desativado.
- **2º Passo** : instalar o upgrade já copiado para o diretório `/score/altcba` do HotStandby pelo comando ***install***:
  - `# cd /`**
  - `# install -u /score/altscore/altcbavxxxx_ddmmaaaa.tgz`**
  - `# install -u /score/altcba/altrcbav0015_25092012.tgz`**
- **3º Passo** : instalar nova tarefa residente pelo comando ***InstalaCtrl***
- **4º Passo** : desativar o micro que está controlando e partir o micro Hot-Standby com a nova versão para controlar a redução
- **5º Passo** : verificar se a nova versão do sistema está funcionando normalmente, movimentação de anodo, EA, corrida, etc...

# Preparação e Instalação de Upgrades

---

## Instalação do Upgrade no Hot-Standby

- **Observações :**

- O Hot-Standby deve ser desativado para instalação somente quando o upgrade envolver **atualização ou inclusão de nova tarefa residente**. Não há necessidade de desativar o controle para instalar programas de operação e relatórios.
- Na operação acima, os dados de base de dados, eventos, históricos e logs ocorridos no período entre a desativação e reativação do Hot-Standby serão perdidos.  
Uma forma de evitar que isso aconteça é reativar o Hot-Standby novamente como reserva e fazer a transferência depois de concluída a atualização dos dados. Nesse caso, se o upgrade envolver qualquer arquivo do diretório [/score/base\\_dados](#), este deve ser atualizado manualmente.
- Nenhum micro de supervisão deve acessar o micro de controle com a versão desatualizada.

# Preparação e Instalação de Upgrades

---

## Instalação do Upgrade nos Nodos de Supervisão

- **1º Passo** : desativar o micro de supervisão e abrir uma janela de comandos do shell do QNX.
- **2º Passo** : copiar o upgrade para seu diretório */score/altcba*
- **3º Passo** : instalar o upgrade pelo comando *install*:
  - # cd /
  - # install -u /score/altscore/altcbavxxxx\_ddmmaaaa.tgz
  - # install -u /score/altcba/altrcbav0015\_25092012.tgz
- **4º Passo** : partir o sistema de supervisão depois que o controle já estiver rodando com a nova versão do sistema



# Preparação e Instalação de Upgrades

---

## Instalação do Upgrade nos Demais Nodos da Fábrica

- **1º Passo** : instalar o upgrade no restante dos nodos da redução testada seguindo os respectivos procedimentos dos micros de supervisão e controle
  - Instalar no outro micro de controle e partir como Hot-Standby
  - Instalar no restante dos micros de supervisão
- **2º Passo** : instalar o upgrade em cada uma das outras reduções seguindo o mesmo procedimento da 1ª redução:
  - Fazer backups da base de dados e runtime antes da instalação
  - Instalar no micro Hot-Standby e num micro de supervisão e acompanhar por um período
  - Instalar no restante dos micros de controle e supervisão
- **3º Passo** : atualizar os backups de todas reduções após atualização
  - Fazer backups da base de dados e runtime de cada redução
  - Fazer backup do micro de desenvolvimento

# Preparação e Instalação de Upgrades

---

## Comandos para Verificação dos Upgrades

- A integridade do arquivo de upgrade com extensão **.F** como *path\_arq.F* pode ser verificados pelo comando **fcats**, conforme a sintaxe abaixo:
- A integridade do arquivo de upgrade com extensão **.tgz** como *path\_arq.tgz* pode ser verificados pelo comando **zcat**, conforme a sintaxe abaixo:

```
# fcats path_arq.F | pax [-v]
```

```
# fcats /score/altcba/altrcbav0015_25092012.F -v
```

```
# zcat path_arq.tgz | pax [-v]
```

```
# zcat /score/altcba/altrcbav0015_25092012.F -v
```

# Preparação e Instalação de Upgrades

---

## Laboratório

- Ler o arquivo de upgrade preparado no laboratório anterior

Apresentação

Ferramentas de Desenvolvimento

Arquitetura do Sistema

Estruturas de Dados

Algoritmo de Controle

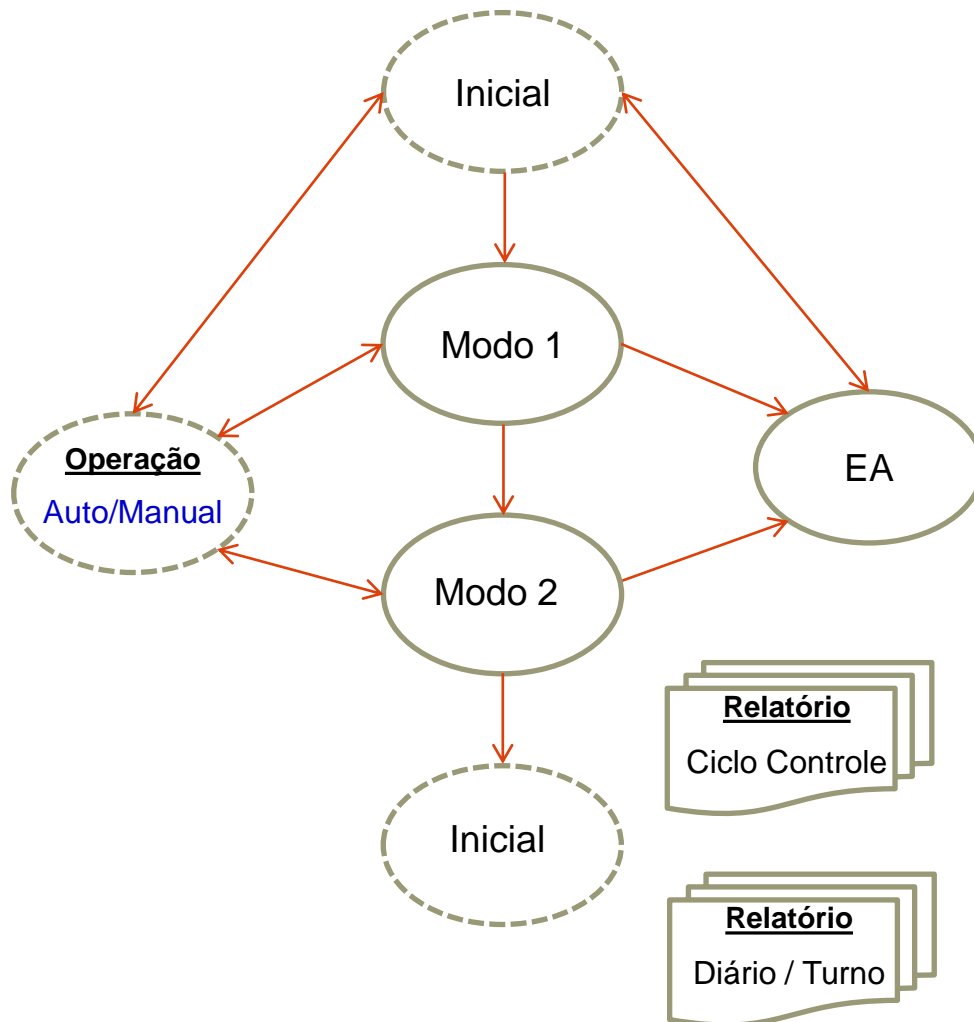
Criação de Opção de Operação

Criação de Relatórios

Preparação e Instalação de Upgrades

**Projeto Final**

## Laboratório – Especificação Algoritmo de Alimentação



- Especificação Funcional:
  - **Startup Sistema**
    - Alimentação Inicial
  - **Operação Manual**
    - **Início:** Suspende Alimentação
    - **Fim:** Volta Alimentação Anterior
  - **Efeito Anódico**
    - **Início:** Alimentação EA
    - **Fim:** Volta Alimentação Inicial
  - **Modos de Operação:**
    - Inicial
    - Modo 1
    - Modo 2
    - EA
    - Suspensa

## Laboratório – Especificação Algoritmo de Alimentação

- Tarefa Residente MCP
  - Implementa mudança de estados
  - **Observação:** O MCP não pode aguardar retorno de comando da remota
- Programa de Operação
  - **Entrada:** Auto / Manual
  - **Ação:** Suspende Alimentação
  - Geração de evento

## Laboratório – Especificação Algoritmo de Alimentação

- Relatório de Ciclo de Controle

### Relatório de Ciclo de Controle de Alimentação

	-Inicial--	-Modo 1-	-Modo 2-	---E A---	-----Modo Atual-----
Cuba	Dur Kg	Dur Kg	Dur Kg	Dur Kg	dd/mm/aa hh:mm Modo
xxx+	xx:xx xxx	xx:xx xxx	xx:xx xxx	xx:xx xxx	xx/xx/xxxx xx:xx xxxxxxxx

Med

- Relatório Diário / Turno

### Relatório Histórico de Alimentação – Turno x

	-Inicial--	-Modo 1-	-Modo 2-	---E A---	---Total---
Cuba	Dur Kg	Dur Kg	Dur Kg	Dur Kg	Dur Kg
xxx+	xx:xx xxx	xx:xx xxx	xx:xx xxx	xx:xx xxx	xxx:xx xxxx

Med

## Laboratório – Implementação Algoritmo de Alimentação

- **1ª Fase – Implementação das Tarefas Residentes**

- Criação e configuração de parâmetros e variáveis de processo
- Implementação da tarefa Mcp
- Implementação da função da tarefa Mcc
- Configuração de eventos (DescrEv e Help)
- Depuração e instalação no controle

- **2ª Fase – Implementação do Programa de Operação**

- Implementação do programa de operação
- Configuração da chamada do programa
- Depuração e instalação no controle



## Laboratório – Implementação Algoritmo de Alimentação

- **3ª Fase – Implementação Relatório de Ciclo de Controle**
  - Criação das variáveis e inclusão dos cálculos nas tarefas
  - Implementação do relatório
  - Configuração dos descritores
  - Criação do Help do relatório
  - Configuração da chamada do relatório
  - Depuração e instalação no controle
- **4ª Fase – Implementação Relatório Diário / Turno**
  - Criação das variáveis e inclusão dos cálculos nas tarefas
  - Implementação do relatório
  - Configuração dos descritores
  - Criação do Help do relatório
  - Configuração da chamada do relatório
  - Depuração e instalação no controle