

Introdução

Boot do QNX

- Partida do sistema operacional:
 1. Seleção da partição de boot na mensagem abaixo:
QNX Loader Boot partition (a digit):
 2. Seleção de boot Normal ou Alternativo pela mensagem abaixo:
Press Esc to Boot alternate OS
 3. Abertura de sessão de usuário (login):
login:
password:
- Opções de boot pelo usuário:
 - Teclas de Comando: **Ctrl+Alt+Shift+Del**
 - Prompt de Comando:
shutdown

Copyright © 2012 Accenture. All rights reserved.

4

Introdução

Diretórios e Processos do Sistema Operacional

- Principais diretórios

/.bitmap	▶ tabela de alocação do sistema
/.licenses	▶ licenças do QNX
/bin	▶ comandos do QNX
/usr/bin	▶ comandos do usuário e QNX
/etc	▶ inicialização de diversos arquivos
/etc/config	▶ arquivos de configuração, sysinit
/tmp	▶ arquivos temporários
- Principais processos do sistema operacional QNX

Proc32	▶ escalonador de processos
Dev	▶ gerenciador de dispositivos
Dev.xxx	▶ drivers de dispositivos (Ex: Dev.con, Dev.par)
Fsys	▶ gerenciador de sistemas de arquivos
Fsys.xxx	▶ drivers de dispositivos de armazenamento
Net	▶ gerenciador de rede
Net.xxx	▶ drivers de placas de rede

Copyright © 2012 Accenture. All rights reserved.

5

Introdução

Laboratório

- Instalar o **Oracle Virtual Toolbox** e criar uma máquina virtual
- Configurar a máquina virtual para ler a imagem .iso do instalador **QNX Product Suite**
- Carregar o **QNX Product Suite** e instalar passo-a-passo:
 1. Tocar numa tecla, selecionar opção **F2: VGA 16 color only** e **<space>**
 2. Selecionar **Next F2** para reconhecer e configurar o hardware
 3. Selecionar **Next F2** e configurar região
Country: Brasil
Keyboard: US - English
Language: English
 4. Selecionar **F4 I Agree** e aceitar os termos da licença
 5. Selecionar **Next F2 – Create Partition** e criar a partição QNX
 6. Configurar o nodo e confirmar instalação completa
New Install
QNX Node Number: 1

Copyright © 2012 Accenture. All rights reserved.

6

Introdução

Laboratório

- Instalação QNX (Continuação) ...
 7. Adicionar as Licenças no campo **Enter License Number** da tela **Add Licenses** teclando **Add / F5** após a digitação de cada licença:
 - qnx.00228405.0j10.816i.a770.0031.50i8
 - tcprt.00461083.0106.gzb4.b300.yjb8.19e1
 - phrt.00413873.0va0.vbd0.m80q.21u3.0xf5
 8. Selecionar opção **Start / F2** para instalar o QNX
 9. Definir boot QNX em modo texto:
Do you want to boot directly into Photon? No
 10. Cancelar configuração de vídeo do Photon:
Do you want to probe and set video modes now? No
 11. Configurar a **QNX4_VM-1** para rede interna

Copyright © 2012 Accenture All rights reserved.

7

Introdução

Laboratório

- Instalação QNX (Continuação) ...
 11. Consultar endereço **IPv4 Address** e **Subnet Mask** atribuído à **QNX4_VM-1** pelo comando **ipconfig**
 12. Configurar IP Address da máquina virtual QNX com um endereço dessa faixa:
 1. Configurar partida do TCP/IP Server:
TCP/IP Mode: Enable TCP/IP Server
Server Setup Configuration: Default
 2. Configurar endereço IP
IP Addresses / This Machine: um endereço IP dessa faixa
NetMask: mesma Subnet Mask consultada pelo ipconfig.
 3. Anotar esse **IP Address** e **NetMask** para laboratório futuro
 13. Selecionar opção **Finish / F3**
 14. Selecionar opção **Reboot F3** para dar boot no QNX

Copyright © 2012 Accenture All rights reserved.

8

Índice

Introdução

Comandos Básicos

Configurações Avançadas

Ferramentas e Desenvolvimento

Controle de Processos

Comunicação entre Processos

Compartilhamento de Informações

Mecanismos de Entrada e Saída

Funções de Tempo

Copyright © 2012 Accenture All rights reserved.

9

Comandos Básicos

Comandos, Diretórios e Arquivos

• Sintaxe e prompt de comandos

```
# cmd -part1 compl    ► comando e parâmetros precedidos de sinal -
# use cmd              ► exibe a sintaxe e parâmetros do comando cmd
                        $ ► prompt de usuário comum
                        # ► prompt de super usuário (root)
# su                   ► sessão de super usuário no shell
# cmd | more           ► pagina a informação exibida na tela
```

• Manipulação de arquivos e diretórios

```
# mv -v path1 path2    ► renomeia ou move path1 para path2
# mkdir path1           ► cria o diretório path1
# rmdir path1           ► apaga o diretório path1
# cp -vpOnr path1 path2 ► copia diretório path1 para path2
# rm -vr path1          ► apaga o diretório path1 e subdiretórios
# pwd                  ► informa o diretório corrente
```

Copyright © 2012 Accenture. All rights reserved.

10

Comandos Básicos

Processos e Sistema de Arquivos

• Manipulação de arquivos

```
# cat file1           ► exibe o arquivo file1
# cp -vp file1 file2   ► copia o arquivo file1 para o file2
# rm -v file1          ► apaga o arquivo;
# zap file1            ► apaga o arquivo corrompido file1
# sort file1           ► ordena o arquivo;
# find path1 -name file1 ► procura o arquivo dentro
# grep pattern file1    ► procura o padrão de nome pattern no arquivo file1
# install -u /path1.tgz ► instala upgrade path1.tgz, vale para extensão .F e .tgz
```

• Espaço em Disco

```
# df -h              ► área livre em disco
# du -k path1        ► área usada pelo diretório path1
# ls -lh             ► lista arquivos e sua área
# ls -lh path1       ► lista arquivos do diretório path1
```

Copyright © 2012 Accenture. All rights reserved.

11

Comandos Básicos

Permissões e Usuários

• Criação de usuário:

```
# passwd user1    ► cria o usuário user1
```

• Criação de grupo:

```
# newgrp group1    ► cria o grupo group1
```

• Alteração do grupo do arquivo:

```
# chgrp [-R] group file1 ► altera o grupo do arquivo file1 e subdiretórios (-R) para group
```

• Alteração do dono do arquivo:

```
# chown [-R] newowner file1 ► altera o dono do arquivo file1 e subdiretórios (-R) para newowner
```

• Alteração de permissões de arquivo:

```
# chmod [-R] mode file1 ► altera permissões do arquivo file1 para os atributos sendo: u-owner, g-group, o-other, a=u+g+o
+ acrescenta permissão, - retira permissão
permissões: r-read, w-write, x-execute
Exemplos Mode: g=rwx, o=f, g+r
```

Copyright © 2012 Accenture. All rights reserved.

12

Comandos Básicos

Laboratório

- Verificar a sintaxe completa do comando `ls`
- Listar arquivos do diretório `/etc`
- Listar arquivos do diretório `/etc` mostrando a data paginando
- Mostrar a lista acima ordenada pelo path
- Listar o path dos arquivos e diretórios de `/etc`
- Listar somente os diretórios de `/etc`
- Localizar o arquivo `netmap` no disco
- Incluir o comando `ls /etc/` no arquivo `/tmp/cmd` e executar:
 - Utilizando o comando `sh`
 - Alterar o atributo para executável e executar novamente

Copyright © 2012 Accenture. All rights reserved.

13

Comandos Básicos

Editor de textos

- Chamada do editor:
 - `# vedit file1` ▶ edita o arquivo `file1`
- Comandos internos do editor:

<code>Ctrl+u</code>	▶ retorna última alteração (undo)
<code>Ctrl+H</code>	▶ apaga linha corrente
<code>F2</code>	▶ passa para próxima ocorrência da palavra
<code>F2</code>	▶ procura palavra
<code>Alt+F2</code>	▶ substituição de palavra
<code>F9</code>	▶ marca início e fim de bloco
<code>Ctrl+F9</code>	▶ copia bloco marcado
<code>Alt+F9</code>	▶ desloca bloco marcado para posição do cursor
<code>Ctrl+F11</code>	▶ copia bloco para área de transferência
<code>F11</code>	▶ copia bloco da área de transferência
<code>Alt x</code>	▶ sair do editor
<code>Alt+F+S</code>	▶ salvar arquivo e sair do editor

Copyright © 2012 Accenture. All rights reserved.

14

Comandos Básicos

Laboratório

- Gerar o arquivo `/tmp/teste` e `/tmp/teste1` com o help dos comandos `ls` e `sin`
- Editar esses arquivos com o `vedit`
 - Apagar uma linha do arquivo teste
 - Copiar um bloco de linhas dentro do mesmo arquivo
 - Copiar um bloco de linhas de um arquivo para o outro
 - Pesquisar uma palavra no arquivo
 - Alterar o fundo da tela
 - Colocar a linha do cursor em highlight.
 - Criar uma macro
- Editar o arquivo `/home/.profile` e incluir
 - **Prompt:** `Nodo.Usuario:Path_Atual`
 - Incluir o path atual no caminho de busca

Copyright © 2012 Accenture. All rights reserved.

15

Índice

Introdução
Comandos Básicos
Configurações Avançadas
Ferramentas e Desenvolvimento
Controle de Processos
Comunicação entre Processos
Compartilhamento de Informações
Mecanismos de Entrada e Saída
Funções de Tempo

Copyright © 2012 Accenture. All rights reserved.

16

Configurações Avançadas

Consoles e Nodos de Rede

- Múltiplas consoles
 - Nº de consoles definido no comando **tinit** do arquivo **sysinit**
 - O sistema Score utiliza as consoles de 1 a 3
 - Comandos de comutação de console (modo texto e photon)
 - CTRL + ALT + ▶ comuta para a console seguinte
 - CTRL + ALT - ▶ comuta para a console anterior
 - CTRL + ALT 3 ▶ comuta para a 3ª console
- Numeração dos nodos da rede
 - Cada nodo é identificado por um número
 - A numeração dos nodos está limitada de 1 a 255
 - A representação dos nodos é precedida de //, por exemplo: //1, //27

Copyright © 2012 Accenture. All rights reserved.

17

Configurações Avançadas

Boot do QNX

1. Seleção de boot Normal ou Alternativo pela mensagem abaixo:

Press Esc to Boot alternate OS

1. Boot Normal :
 - .boot** // Imagem de boot (módulo executável)
 - /etc/config/sysinit.\$Nid** // arquivo interpretado com sufixo nº do nodo
 - ou
 - /etc/config/sysinit** // executado quando SO não encontra o arquivo com sufixo nº do nodo.
2. Boot Alternativo : // servidor photon reserva
 - .altboot** // Imagem de boot alternativo (módulo executável)
 - /etc/config/altsysinit** // arquivo inicialização de boot alternativo
2. Abertura de sessão de usuário (login): // comando **tinit (sysinit)**
 - login:**
 - password:**

Copyright © 2012 Accenture. All rights reserved.

18

Configurações Avançadas

Boot QNX – Imagem de Boot

- Principais características
 - Somente comandos que serão pouco alterados
 - Módulo Executável (compilado e linkado)
- Principais diretórios:
 - /boot/build** // arquivo fonte
 - /boot/images** // arquivo executável (imagem)
 - /boot** // cópia da imagem executada no boot
- Principais processos
 - # Proc -l \$Nid** ▶ gerenciador de processos (1º processo)
 - # Slib** ▶ gerenciador de bibliotecas compartilhadas
 - # Fsys -f 4000** ▶ gerenciador do sistema de arquivos
cria *ramdisk* de 4 Mb
 - # mount -p \$h \$ht77** ▶ monta o HD
 - # Net** ▶ gerenciador de rede

Copyright © 2012 Accenture. All rights reserved.

19

Configurações Avançadas

Boot QNX – Arquivo Sysinit

- Principais diretórios
 - /etc/config/sysinit.\$Node** // arquivo fonte
 - /etc/config/sysinit.score.\$Node** // arquivo executável (imagem)
 - /etc/config/altsysinit** // cópia da imagem executada no boot
- Principais inicializações
 - Relógio
 - Drivers de dispositivos
 - Montagem de dispositivos e *ramdisk*
 - Inicialização de rede
 - Gerenciador de filas
 - Gerenciador do circuito virtual
 - Comando *tinix*
 - Configuração do número de consoles
 - Partida automática do Score
 - Partida do *login*

Copyright © 2012 Accenture. All rights reserved.

20

Configurações Avançadas

Laboratório

- Configurar a máquina virtual de desenvolvimento com as seguintes características
 - Nodo 1 para boot normal
 - Nodo 4 para boot alternativo
- Replicar a máquina Virtual QNX4_VM-1 para QNX4_VM-2 e QNX4_VM-3
- Configurar as máquinas virtuais QNX4_VM-2 e QNX4_VM-3 como nodos 2 e 3 respectivamente

Copyright © 2012 Accenture. All rights reserved.

21

Configurações Avançadas

Administração de Rede

- Arquivo de configuração de Rede

- Path:** `/etc/config/netmap`
- Descrição:** mapeamento dos nodos QNX
- Formato do Arquivo:**

NodeId	NetId	MacAddress
1	1	0000c04a9330
1	2	0000c04a9130
2	1	0000c04a9320
2	2	0000c04a9331
3	1	0000c04a9430
4	1	0000c04a9355

► **NodeId:** Nº do nodo
 ► **NetId:** Nº da subrede
 ► **MacAddress:** Endereço da placa de rede

- Partida do driver de rede e inicialização do netmap em memória

```
# nettrap start
# netmap -f
```

Copyright © 2012 Accenture. All rights reserved.

22

Configurações Avançadas

Administração de Rede

- Principais Processos de Rede (Sysinit / Altsysinit)

# Net	► gerenciador de rede
# Net.rtl	► driver da placa de rede chipset Realtek
# Socket	► gerenciador TCP/IP
# nameioc	► gerenciador circuito virtual

- Referências Técnicas:

- Guia de Manutenção e Instalação:** Capítulo 1 – Arquitetura de Rede do Score
- Manuais QNX**

Copyright © 2012 Accenture. All rights reserved.

23

Configurações Avançadas

Laboratório

- Colocar as máquinas virtuais QNX4_VM-1, QNX4_VM-2, QNX4_VM-3 em rede
- Configurar o micro de desenvolvimento da seguinte forma:
 - Boot Normal: fora da rede
 - Boot Alternativo: em rede com as 3 máquinas virtuais QNX4_VM-1, QNX4_VM-2 e QNX4_VM-3

Copyright © 2012 Accenture. All rights reserved.

24

Configurações Avançadas

Administração de Rede

- Arquivos de configuração TCP/IP
 - `/etc/config/hosts`: mapeamento de dispositivos da rede TCP/IP
 - `/etc/config/netstart`: partida do serviço TCP/IP
- Principais comandos TCP/IP

<code># ifconfig [Interface]</code>	▶ verifica endereço IP do micro
<code># ifconfig en1</code>	▶ verifica endereço IP da 1ª interface ethernet
<code># netstart</code>	▶ ativa o serviço TCP/IP, configurado na instalação do Score (Score.init)

Copyright © 2012 Accenture. All rights reserved.

25

Configurações Avançadas

Comandos de Manutenção

- Verificação do disco

<code># fdisk /dev/hd0</code>	// verifica partições do disco
<code># chksys -R /dev/hd0t77</code>	// verifica e repara integridade do HD
- Teste de comunicação da porta serial

<code># qtalk -m /dev/ser1</code>	// conecta à porta serial 1
<code># qtalk -m //13/dev/ser2</code>	// conecta à porta serial 2 do nodo 13

 - **Comando Saída:** CTRL + A + q
- Acesso remoto

<code># on -n 13 -t /dev/con4 login</code>	// executa login na console 4 do nodo 13
<code># on -n 3 -r //3/ make</code>	// executa make no nodo 3 com path raiz no próprio nodo 3
<code># ditto -n 13 -t 4 -k</code>	// conecta à console 4 do nodo 13

 - **Comando Saída:** CTRL + e + q

Copyright © 2012 Accenture. All rights reserved.

26

Configurações Avançadas

Laboratório

- Partir as máquina virtuais QNX4_VM-1, QNX4_VM-2 e QNX4_VM-3 em rede
 - Logar na console 5 de QNX4_VM-2 via ditto pela QNX4_VM-1)
- Salvar as configurações Atuais:
 - Fazer backup das máquinas virtuais (nodos 1, 2 e 3)
 - Gerar boot alternativo igual ao principal
 - Salvar arquivos `netmap` e `sysinit`
- Instalar o sistema Score nos nodos 1, 2 e 3 a partir do micro de desenvolvimento:
 - Score Runtime
 - Base de Dados de Demonstração
- Partir o sistema e ver o que acontece
- Colocar os micros em rede novamente

Copyright © 2012 Accenture. All rights reserved.

27

Configurações Avançadas

Comandos de Data e Hora

- Exibir data e hora na console
 - # clock & ▶ exibe data e hora no canto superior esquerdo da console
 - # clock -b black -f yellow &
- Verificar e alterar data e hora (memória)
 - # date ▶ exibe a data atual
 - # date 22 11 12 ▶ altera a data para 22/11/2012
- Atualiza e lê hora do relógio de tempo real
 - # rtc hw ▶ exibe data do relógio de tempo real do computador
 - # rtc -s hw ▶ altera a data do relógio de tempo real a partir da data da memória
 - # rtc net 5 ▶ exibe data do relógio de tempo real do nodo 5
 - # rtc -sl hw net 5 ▶ atualiza a data do relógio de tempo real do micro local a com a data do nodo 5 (memória)

Copyright © 2012 Accenture. All rights reserved.

28

Configurações Avançadas

Laboratório

- Testar os comandos de relógio :
 - Alterar a data e hora na QNX4_VM-1 e atualizar QNX4_VM-2 através dela

Copyright © 2012 Accenture. All rights reserved.

29

Índice

Introdução

Comandos Básicos

Configurações Avançadas

Ferramentas e Desenvolvimento

Controle de Processos

Comunicação entre Processos

Compartilhamento de Informações

Mecanismos de Entrada e Saída

Funções de Tempo

Copyright © 2012 Accenture. All rights reserved.

30

Ferramentas e Desenvolvimento

Programação em linguagem C

- Sistema desenvolvido em linguagem C e C++
- Estrutura programa


```
# include "stdio.h" ou <stdio.h>
Protótipo funções e procedimentos
main program(int argc, *char Argv[])
{
    Corpo programa principal
}
Funções e procedimentos
Estrutura programa
```
- Programa executável (Modular)
 1. Compilação e Linkedição ► wcc (módulo objeto e executável)
 2. Bibliotecas ► wlib (biblioteca de módulos objeto)

Copyright © 2012 Accenture. All rights reserved.

31

Ferramentas e Desenvolvimento

Makefile

- Principais características
 - Declaração de dependências arquivos .h (prólogos/includes), .o (módulo objeto), .c (fontes) e .lib (bibliotecas)
 - Compilação e linkedição somente de arquivos necessários
 - A relação de dependência do sistema depende do makefile
- Formato makefile
 - Declaração de bibliotecas e prólogos


```
LIBS: Photon.lib RedeRem.lib
HDRS: PhHdr.h RemHdr.h
```
 - Dependência do módulo executável (fontes e prólogos ou includes)


```
MyTrgt1: MySrc1.o $(LIBS)
```
 - Dependência de módulo objeto


```
MySrc1.o: MySrc1.c MyHdr.h $(HDRS)
          wcc MySrc1.c
```

Copyright © 2012 Accenture. All rights reserved.

32

Ferramentas e Desenvolvimento

Ferramentas de Desenvolvimento

- Principais componentes do compilador Watcom C/C++
 - # wcc ► compilador e linkeditor, gera módulos objeto e executáveis
 - # wlib ► cria e consulta bibliotecas de módulos objeto
 - # wtrip ► elimina informações de debug do módulo objeto tornando o tamanho do arquivo menor
 - # wd ► debugger de programas

Copyright © 2012 Accenture. All rights reserved.

33

Ferramentas e Desenvolvimento

Laboratório

- Abrir o makefile de geração de imagem de boot e analisar, identificando os seguintes itens:
 - # Declaração de includes e bibliotecas
 - # Dependência de fontes, includes, objetos e bibliotecas
 - # Passagens de parâmetros como macros
 - # Labels

Copyright © 2012 Accenture All rights reserved.

34

Ferramentas e Desenvolvimento

Interface Photon

- Chamada do Photon
 - # `ph`
- Principais recursos do Photon para desenvolvimento
 - Help
 - Sin Gráfico
 - Shell (pterm)
 - Aplicações de desenvolvimento Score em ambiente Photon
- Arquivo de configuração de video:
 - `/etc/config/trap/crt.$NODE`
- Uso do Phindows para desenvolvimento

Copyright © 2012 Accenture All rights reserved.

35

Ferramentas e Desenvolvimento

Preparação da Plataforma de Desenvolvimento

- Partir a interface photon e conhecer
 - # `ph`
- Instalar o compilador Watcom conforme a sequencia abaixo:
 1. Instalar as licenças do compilador
 2. Instalar o compilador Watcom pelo cd QNX Product Suite pelo comando `phinstall`
 3. Instalar o upgrade `curso_qnx.tgz` com os laboratórios do curso QNX
- Configurar o video do photon no nodo 3 para partir o sistema
 - Partir como boot alternativo e desativar a partida automática
 - Apagar o arquivo `/etc/config/trap/crt.$NODE`
 - Configurar o vídeo para 1024 x 768

Copyright © 2012 Accenture All rights reserved.

36

Configurações Avançadas

Programação Shell - Introdução

- Chamada do arquivo de comandos **cmdshell** pelo utilizando o shell ou não :


```
# sh cmdshell           // arquivo texto comum
# cmdshell              // precisa mudar o atributo para executável
# chmod a+x cmdshell    // converte o arquivo para executável
```
- Variáveis de Ambiente:
 - O QNX possui várias variáveis de ambiente cujo nome é sempre em letras maiúsculas com prefixo \$, exemplo: **\$NODE**
 - As principais variáveis de ambiente são atribuídas dentro do arquivo **.profile** para o usuário e pelo comando **export** para o sistema operacional.
- Sintaxe:
 - O caractere **#** é a marca de comentário no arquivo shell

Copyright © 2012 Accenture. All rights reserved.

37

Configurações Avançadas

Programação Shell – Entrada e Saída

- Leitura de linha de comandos pelos comandos **read** e **echo** dentro do arquivo de comandos shell:


```
# cmdshell Hello world! // chamada do arquivo shell
# read v1                # nesse caso v1= "Hello world!"
# echo $v1               # exibe o valor de v1, ou seja "Hello world!"
```
- Passagem de parâmetros:
 - **Posição:** parâmetros separados por espaço na chamada do programa


```
# copia arq1 arq2      // dentro do programa: arq1 = $1 e arq2 = $2
# echo $1 e $2          # exibe os valores dos parâmetros passados
```
 - **Referência:** atribui uma variável na chamada


```
# gmake b=hard.1       // dentro do programa o parâmetro: b = $b
# echo $b              # exibe o valor do parâmetro
```

Copyright © 2012 Accenture. All rights reserved.

38

Ferramentas e Desenvolvimento

Laboratório

- Criar um arquivo de comandos **cmdtest** e incluir:
 - Exibição da mensagem **"Mensagem do Shell"**
 - Exibição do 1º parâmetro
- Executar o comando
 - Chamar o comando direto no prompt
 - Executar o comando utilizando o comando **sh**
 - Alterar os atributos de execução e executar sem o comando **sh**

Copyright © 2012 Accenture. All rights reserved.

39

Ferramentas e Desenvolvimento

Programação Shell – Estruturas de Programação

- Comandos **if**, **elif** (else if) e **else**

```
if expr1 then
  list1
[elif expr2 then
  list2] ...
[else
  list3]
fi
```

- Expressões de testes numéricos

```
V1 -eq V2    # V1 = V2
V1 -ne V2    # V1 <> V2
V1 -ge V2    # V1 >= V2
V1 -gt V2    # V1 > V2
V1 -le V2    # V1 <= V2
V1 -lt V2    # V1 < V2

string1 = string2
string1 != string2
```

Copyright © 2012 Accenture. All rights reserved.

40

Ferramentas e Desenvolvimento

Programação Shell – Estruturas de Programação

- Atribuição de valores

```
Let var1 = 100    # atribui 100 a var1
```

- Comando **Case**

```
case word in
  pattern1)      list1;;
  pattern2 | pattern1) list2;;
  ...
  *)             listdefault;;
esac
```

- Comando **For**

```
For name in list
do
  list1
done
```

Copyright © 2012 Accenture. All rights reserved.

41

Ferramentas e Desenvolvimento

Programação Shell – Estruturas de Programação

- Comando **While**

```
While test $var1 -lt 100
do
  echo $a
  let var1 = var1 + 1
done
```

- Comando **Until**

```
Var1 = 0
Until test $var1 -eq 100
do
  echo $var1
  let var1 = var1 + 1
done
```

- Comandos de Interrupção de Loop (**for**, **while**, **until**)

- break**: interrompe o loop e passa para o 1º comando fora dele
- continue**: retorna ao início da iteração seguinte do loop

Copyright © 2012 Accenture. All rights reserved.

42

Ferramentas e Desenvolvimento

Programação Shell – Estruturas de Programação

- Criação de função **Funcshell**
 - Declaração da função
Function Funcshell ou **Funcshell ()**

```
{
  lista
}
```
 - Chamada da função **Funcshell**
Funcshell
- Arquivos de comando shell importantes:
 - **QNX**: sysinit, sysinit.\$NODE, altsysinit, .profile
 - **Score**: setup, netstart, ManutRede

Copyright © 2012 Accenture. All rights reserved.

43

Ferramentas e Desenvolvimento

Laboratório

- Implementar um arquivo de comandos para montar um pendrive USB com as seguintes características :
 - **Sem parâmetros** : monta dispositivo QNX
 - **Parâmetro -q**: monta dispositivo QNX
 - **Parâmetro -d**: monta dispositivo DOS
 - Consistir parâmetros e emitir mensagem de uso em caso de erro
- Localizar e editar o arquivo de **setup** de instalação do sistema em QNX4_MV-4 e identificar:
 - Geração do arquivo de boot **hard.\$NODE**
 - Quais nodos tem TCP/IP
 - Instalação do arquivo **queue**
- Localizar e editar o arquivo **ManutRede** e identificar:
 - Estruturas de programação case, if, read, case e utilização de funções
 - Redirecionamento de nodo de trabalho.
 - Diferença de tratamento entre os nodos de controle e supervisão

Copyright © 2012 Accenture. All rights reserved.

44

Índice

Introdução

Comandos Básicos

Configurações Avançadas

Ferramentas e Desenvolvimento

Controle de Processos

Comunicação entre Processos

Compartilhamento de Informações

Mecanismos de Entrada e Saída

Funções de Tempo

Copyright © 2012 Accenture. All rights reserved.

45

Controle de Processos

Multiprocessamento

- Principais características:
 - Multi-Usuário e Multi-Tarefa
 - Suporte a sistemas modulares
 - Comunicação e sincronismo entre tarefas
 - Operações em background e foreground simultaneamente
 - Prioridade de execução de tarefas (1 a 30)
- Processos de gerenciamento do sistema operacional
 - # Proc ► controle de todos recursos usados pelos processos
 - # Fsys ► operações de I/O de discos
 - # Dev ► controle de dispositivos de I/O
 - # Net ► gerenciamento da rede de comunicação
 - # idle ► preenche o tempo livre do processador
 - # queue ► controle de filas de comunicação
 - # nameloc ► administra o circuito virtual de rede

Copyright © 2012 Accenture. All rights reserved.

46

Controle de Processos

Status e Comandos

- Principais estados dos processos:
 - Ready ► pronto para rodar (escalonados de acordo com a prioridade)
 - Blocked ► bloqueado por *send*, *receive*, *reply* ou *signal*
 - Held ► preso pela interrupção SIGSTOP
 - Wait ► aguardando a execução da tarefa filha criada
 - Dead ► tarefa *Zombie*, tarefa morreu sem enviar *send* ao criador
- Escalonamento de tarefas (Adaptativo, Round Robin, Fifo)
- Executar o comando *sin* e discutir
 - SID (section id)
 - Pid (process id)
 - PRi (prioridade)
 - STATE
 - BLK

Copyright © 2012 Accenture. All rights reserved.

47

Controle de Processos

Status e Comandos

- Cuidado! Evitar Loop Ativo
- Alguns comandos de processos
 - # sin ► exibe processos em execução
 - # slay *process* ► mata tarefa *process* em execução
 - # kill -p 1023 ► mata tarefa em execução de PID 1023
 - # sac -i 10 -p 20 ► mostra taxa de uso da CPU

Copyright © 2012 Accenture. All rights reserved.

48

Controle de Processos

Laboratório

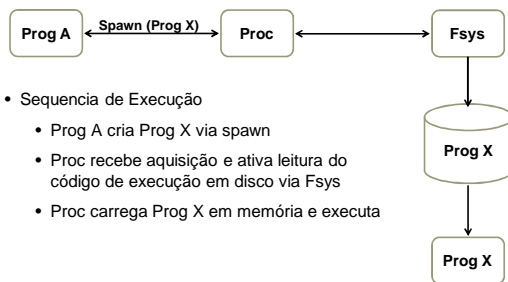
- Abrir o vedit e verificar o status das tarefas
 - Verificar a hierarquia de criação das tarefas
 - Matar a tarefa pai do vedit e verificar o que aconteceu

Copyright © 2012 Accenture. All rights reserved.

49

Controle de Processos

Criação de Tarefas



- Sequencia de Execução
 - Prog A cria Prog X via spawn
 - Proc recebe aquisição e ativa leitura do código de execução em disco via Fsys
 - Proc carrega Prog X em memória e executa

Copyright © 2012 Accenture. All rights reserved.

50

Controle de Processos

Criação de Tarefas

- Se a tarefa criadora morrer todas tarefas filhas também morrem
- Primitivas de criação de tarefas:
 - qnx_spawn(...)**
 - ▶ cria tarefas utilizando toda estrutura de permissões e prioridades do sistema operacional QNX
 - Cria a tarefa filha e suspende a execução do pai até a filha terminar ou de forma que as duas executem de forma concorrente
 - Cria somente módulos executáveis (programas)
 - system(...)**
 - ▶ abre uma instância do shell e executa uma string de comando montada previamente
 - Cria a tarefa filha e suspende a execução do pai até a filha terminar ou de forma que as duas executem de forma concorrente
 - Cria 3 tipos de processos diferentes:
 - Módulos executáveis ou programas
 - Comandos do sistema operacional QNX
 - Shell scripts

Copyright © 2012 Accenture. All rights reserved.

51

Controle de Processos

Laboratório

- Criar makefile para compilar os programas `qspawn.c` e `child.c` no diretório `/curso_qnx/qnxspawn`
- Compilar e executar o programa `qspawn.c` e `child.c` no diretório `/curso_qnx/qnxspawn`
 - Testar com `exec=0` e `exec=1`
 - Alterar prioridade
 - Testar criando filho em outro nodo da rede

Copyright © 2012 Accenture. All rights reserved.

52

Índice

Introdução
Comandos Básicos
Configurações Avançadas
Ferramentas e Desenvolvimento
Controle de Processos
Comunicação entre Processos
Compartilhamento de Informações
Mecanismos de Entrada e Saída
Funções de Tempo

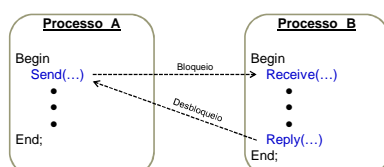
Copyright © 2012 Accenture. All rights reserved.

53

Comunicação entre Processos

Comunicação Síncrona

Bloqueada



- Dead Lock
- Solução
 - Produtor → Consumidor
 - Programa Agente para intermediar
- Exemplos:
 - MCP → MCC
 - MCP → McpCorrida
 - MCP → AePotTerm

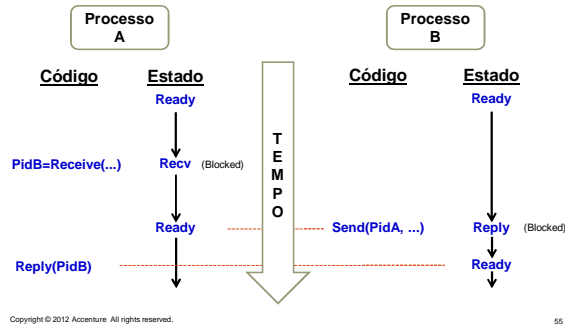
Copyright © 2012 Accenture. All rights reserved.

54

Comunicação entre Processos

Comunicação Síncrona

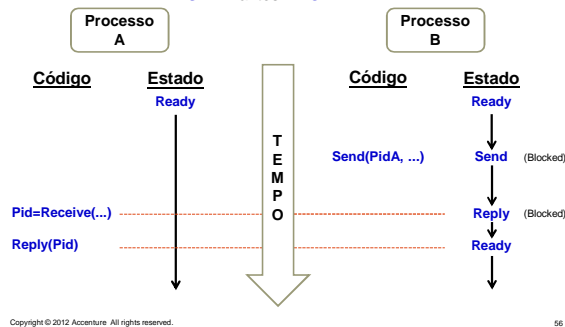
RECEIVE antes SEND



Comunicação entre Processos

Comunicação Síncrona

SEND antes RECEIVE



Comunicação entre Processos

Comunicação Síncrona



- Primitivas de comunicação (QNX)
 - `send(PidB, ...)` ➤ **Tarefa A** envia mensagem para **Tarefa B** (Bloqueio)
 - `PidA=receive(...)` ➤ **Tarefa B** recebe mensagem da **Tarefa A**
 - `reply(PidA,...)` ➤ **Tarefa B** retorna mensagem à **Tarefa A** (Desbloqueio)
- Apresentar as funções abaixo utilizando o help do QNX no photon
 - `Send`
 - `Receive`
 - `Reply`

Comunicação entre Processos

Laboratório

- Compilar e abrir os programas `envia.c` e `recebe.c` do diretório `/curso_qnx/mgs`
- Partir a tarefa recebe:
 - # `Recebe &`
- Executar o programa `envia` passando o `pid` informado pelo `recebe` anterior
- Alterar o programa `recebe` para ver o programa `envia` bloqueado num `reply`
- Alterar o programa `recebe` para o programa `envia` ficar bloqueado num `send`.

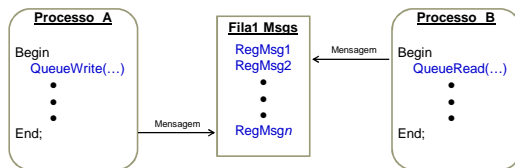
Copyright © 2012 Accenture All rights reserved.

58

Comunicação entre Processos

Comunicação Assíncrona - Queue

Não-Bloqueada (Queue)



- Exemplos:
 - Fila Eventos
 - Fila Comandos ATN8

Copyright © 2012 Accenture All rights reserved.

59

Comunicação entre Processos

Comunicação Assíncrona - Queue



- Administrador de filas
 - # `queue &` ► Gerencia as filas de mensagens
 - Programa customizado pela equipe de desenvolvimento
- Informações mais detalhadas de implementação arquivo abaixo no QNX, instalado pelo kit de desenvolvimento
 - `/score/srcs/queue/README`
- Prólogo de filas
 - # `include <queue.h>`

Copyright © 2012 Accenture All rights reserved.

Comunicação entre Processos

Comunicação Assíncrona - Queue

- Primitivas de criação / fechamento da fila
 - **int queue_open(char *name, char *mode, unsigned window_size);**
 - **Função:** cria e/ou abre a fila de mensagens
 - **Parâmetros:** nome, modo de acesso e tamanho máximo da fila.
 - **Retorno:** ≥ 0 → **id** da fila
 < 0 → código de erro
 - **int queue_close(int id)**
 - **Função:** fecha a fila de mensagens
 - **Parâmetros:** **id** da fila retornado pela função **queue_open**
 - **Retorno:** 0 → fila fechada
 < 0 → código de erro

Copyright © 2012 Accenture. All rights reserved.

Comunicação entre Processos

Comunicação Assíncrona - Queue

- Primitivas de leitura / escrita de mensagens na fila
 - **queue_write(int id, void *msg, int msg_size, int priority)**
 - **Função:** grava mensagens na fila
 - **Parâmetros:** **id** da fila, ponteiro para a mensagem, tamanho da mensagem (máximo 2000 bytes) e prioridade (0 → mais alta)
 - **Retorno:** 0 → mensagem postada
 < 0 → código de erro
 - **int queue_read(int id, int wait, void *msg, int msg_size);**
 - **Função:** lê e retira mensagens da fila
 - **Parâmetros:** **id** da fila, modo de espera da leitura, ponteiro para a mensagem e tamanho da mensagem
 - **Retorno:** 0 → mensagem postada
 < 0 → código de erro

Copyright © 2012 Accenture. All rights reserved.

Comunicação entre Processos

Laboratório

- Compilar os programas do diretório **/curso_qnx/queue**
 - Queueercv – recebe dado da fila
 - Queuesnd – envia dado para a fila
- Executar o programa **queueercv**

```
# queueercv
```
- Executar o programa **queuesnd** em outra janela


```
# Queuesnd "Mensagem enviada"
```
- Analisar o comportamento dos programas

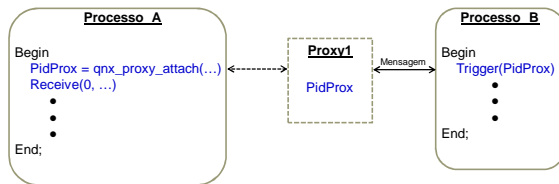
Copyright © 2012 Accenture. All rights reserved.

63

Comunicação entre Processos

Comunicação Não-Bloqueada - Proxy

Não-Bloqueada (Proxies)



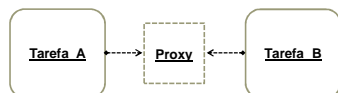
- Exemplos:
 - ATN1.4 → DrvSad (Interrupção Hardware gera um proxy)
 - DrvSad → MCP

Copyright © 2012 Accenture. All rights reserved.

64

Comunicação entre Processos

Comunicação Não-Bloqueada - Proxy



- Primitivas de comunicação (QNX)
 - `PidProxy = qnx_proxy_attach(...)`
 - `PidB = receive(0, ...)`
 - `Trigger(PidProxy, ...)`
- Tarefa A atraca ao proxy
 - Tarefa A pronta para receber mensagem de **qualquer tarefa**
 - Tarefa B envia mensagem para Tarefa A que executa sem bloqueá-la
- Apresentar as funções acima utilizando o help do QNX no photon

Copyright © 2012 Accenture. All rights reserved.

Comunicação entre Processos

Laboratório

- Compilar os programas do diretório `/curso_qnx/proxies`
 - `recprox` – recebe proxy
 - `sendprox` – envia sinal via Pid da tarefa sem mensagem
 - `sendpr2` – atraca proxy no `recprox` e envia sinal pelo `ProxyId` passando ponteiro da mensagem
- Executar o programa `recprox`

```
# Recprox
```
- Verificar proxies criados pelo comando `sin`

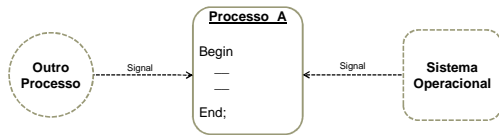
```
# sin proxies
```
- Executar o programa `sendprox` e `sendpr2` em outra janela passando como parâmetro os ids informados
- Analisar o comportamento dos programas

Copyright © 2012 Accenture. All rights reserved.

66

Comunicação entre Processos

Comunicação por Interrupção de Software



- Sinalização para matar ou interromper processos de forma assíncrona
- Apresentar as funções acima utilizando o help do QNX no photon
- Exemplos:
 - Erro Execução (Handler)
 - Comandos SO: slay, kill

Copyright © 2012 Accenture. All rights reserved.

67

Comunicação entre Processos

Laboratório

- Compilar os programas do diretório `/curso_qnx/signals`
 - `killflex` – sinaliza um signal para o processo
 - `sighndl` – trata 2 signals (SIGUSR1 e SIGUSR2)
- Executar o programa `sighndl`

```
# sighndl
```
- Executar o programa `killflex` em outra janela passando como parâmetro o pid e o número do signal tratado (16 e/ou 17). (Consultar `/usr/include/signal.h`)
- Analisar o comportamento dos programas

Copyright © 2012 Accenture. All rights reserved.

68

Comunicação entre Processos

Circuito Virtual (VC)

`nameloc &` ▶ gerenciador do circuito virtual (VC)

- Primitivas de registro nomes no circuito virtual (QNX)
 - `qnx_name_attach(...)` ▶ registra nome na rede (VC)
 - `qnx_name_locate(...)` ▶ localiza nome registrado na rede (VC)
 - `qnx_name_detach(...)` ▶ cancela o nome registrado na rede
 - O registro do nome também é cancelado se a tarefa que registrou morrer
- Aplicações do registro de nomes
 - Implementação de semáforos (AdminSemaf)
 - Verificar se as tarefas residentes estão executando (Watchdog)
- Outras primitivas de rede
 - `getpid(...)` ▶ consulta o pid da própria tarefa
 - `getppid(...)` ▶ consulta o pid da tarefa criadora
 - `getnid(...)` ▶ consulta o nodo de rede da tarefa

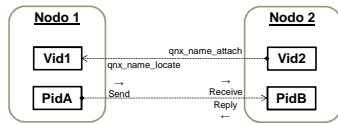
`sin na` ▶ verifica os nomes registrados na rede

Copyright © 2012 Accenture. All rights reserved.

69

Comunicação entre Processos

Comunicação Bloqueada (Rede)



- Primitivas de registro de nome (QNX)
 - `qnx_name_attach(...)` ▶ **Tarefa B** registra nome na rede
 - `PidB=qnx_name_locate(...)` ▶ **Tarefa A** consulta nome, estabelece circuito virtual entre nodos e recebe **Vid2 = PidB**
- Primitivas de comunicação (QNX)
 - `send(PidB, ...)` ▶ **Tarefa A** envia mensagem para **Tarefa B** (Bloqueio)
 - `PidA=receive(...)` ▶ **Tarefa B** recebe mensagem da **Tarefa A**
 - `reply(PidA,...)` ▶ **Tarefa B** retorna mensagem à **Tarefa A** (Desbloqueio)

Copyright © 2012 Accenture. All rights reserved.

Comunicação entre Processos

Laboratório

- Partir o sistema score principal, reserva e supervisão
 - Verificar as tarefas que estão em execução nos micros de controle
 - Desativar o principal e acompanhar a partida do reseva
 - Analisar essa comunicação
- Verificar os nomes que estão registrados na rede pelo comando `sin`

Copyright © 2012 Accenture. All rights reserved.

71

Índice

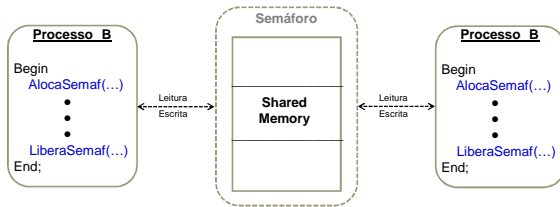
- Introdução
- Comandos Básicos
- Configurações Avançadas
- Ferramentas e Desenvolvimento
- Controle de Processos
- Comunicação entre Processos
- Compartilhamento de Informações**
- Mecanismos de Entrada e Saída
- Funções de Tempo

Copyright © 2012 Accenture. All rights reserved.

72

Compartilhamento de Informações

Memória Compartilhada



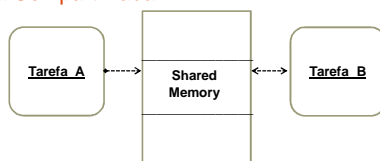
- O acesso à memória compartilhada do Score é feito através de bibliotecas do sistema.
- Exemplos: AVC e Comum

Copyright © 2012 Accenture. All rights reserved.

73

Compartilhamento de Informações

Memória Compartilhada



- Primitivas de criação da memória compartilhada
 - `shm_open(...)` ▶ abre e/ou aloca segmento de memória
 - `ltruncate(...)` ▶ define tamanho do segmento criado
 - `mmap(...)` ▶ monta o segmento de memória para acesso
 - `close(...)` ▶ fecha o segmento criado (não apaga segmento)
 - `shm_unlink(...)` ▶ remove o segmento de memória
- Apresentar as funções acima utilizando o help do QNX no photon

Copyright © 2012 Accenture. All rights reserved.

Compartilhamento de Informações

Laboratório

- Compilar os programas do diretório `/curso_qnx/sharemem`
 - ShmExample1 – exemplo de mapeamento de área física
 - ShmExample2 – exemplo de uma tarefa criando a memória compartilhada (cria, escreve e morre)
 - ShmExample3 – exemplo de mapeamento e criação de 2 tarefas acessando a mesma memória
- Executar o programa `ShmExample1`
- Executar o programa `ShmExample2` com depurador e ver memora aberta (`ls -lh /dev/shmem`)
- Executar o programa `ShmExample3`

Copyright © 2012 Accenture. All rights reserved.

75

Índice

Introdução
Comandos Básicos
Configurações Avançadas
Ferramentas e Desenvolvimento
Controle de Processos
Comunicação entre Processos
Compartilhamento de Informações
Mecanismos de Entrada e Saída
Funções de Tempo

Copyright © 2012 Accenture. All rights reserved.

76

Mecanismos de Entrada e Saída

Operações de I/O

- O ambiente de desenvolvimento QNX permite a implementação de acesso direto à memória e portas de I/O do processador do computador.
- Nesse tópico os recursos serão apenas citados para conhecimento, pois já estão encapsulados nas bibliotecas do Score para o usuário.
- **Memory-Mapped I/O**
 - O mapeamento e acesso direto à memória via programas em C é implementado através do comando `shm_open` conforme exemplo 1 do laboratório anterior.

Copyright © 2012 Accenture. All rights reserved.

77

Mecanismos de Entrada e Saída

Operações de I/O

- **I/O Port**
 - O acesso às portas de comunicação da CPU com os dispositivos de I/O mapeados é possível através dos comandos `inp`, `outp`, `inpw` e `outw`.
 - Um exemplo de utilização de I/O Port no sistema é a comunicação com as placas PCLTA.
- **Serial I/O**
 - A comunicação com porta serial também pode ser implementada via programação C.
 - O QNX a nível de sistema operacional já disponibiliza drivers como `Dev.ser` para comunicação os dispositivos seriais como por exemplo `/dev/ser1` e `/dev/ser2`.
 - Como exemplo temos a comunicação entre os micros de controle principal e reserva via serial e o utilitário `qtalk` do QNX

Copyright © 2012 Accenture. All rights reserved.

78

Mecanismos de Entrada e Saída

Operações de I/O

• Trigger e Proxies

- A comunicação com um dispositivo de hardware pode ser implementada atribuindo um **Proxy** ao dispositivo e posteriormente acionando-o através de seu respectivo **Trigger** associado.
- Um exemplo de utilização de **Trigger** e **Proxy** no sistema é a leitura dos sinais analógicos de corrente e tensão implementada pelo DRVSAD.

• Entrada e Saída padrão

- Conceito utilizado na maioria dos programas no QNX

• Terminal Devices - Consoles

- A comunicação com terminais de consoles como **/dev/stty1**, **/dev/stty2** também é possível através do driver **Dev.stty** redirecionando saída ou através de outras operações, como por exemplo o monitoramento remoto através via **ditto**.

Copyright © 2012 Accenture All rights reserved.

79

Mecanismos de Entrada e Saída

Operações de I/O

• Terminal Devices – Photon

- É possível criar janelas texto de console no ambiente photon através da aplicação **pterm**. A cada uma destas janelas é associado um dispositivo de I/O, como **/dev/ttyp1**, **/dev/ttyp2** etc.

• Interrupções

- O QNX permite também a implementação de interrupções de hardware e software como forma de comunicação.
- Um exemplo de interrupção por software é o tratamento de exceções utilizado para depuração em algumas tarefas.

Copyright © 2012 Accenture All rights reserved.

80

Índice

Introdução

Comandos Básicos

Configurações Avançadas

Ferramentas e Desenvolvimento

Controle de Processos

Comunicação entre Processos

Compartilhamento de Informações

Mecanismos de Entrada e Saída

Funções de Tempo

Copyright © 2012 Accenture All rights reserved.

81

Funções de Tempo

Tratamento de Tempo

- Principais funções de tratamento de tempo
 - time(NULL)** ▶ pega data corrente no formato interno do sistema operacional
 - localtime(&time)** ▶ converte hora no formato interno para *struct tm*
 - mktime(time_tm)** ▶ converte hora no formato *struct tm* para formato interno do sistema operacional
 - sleep(x)** ▶ suspende o processo por x segundos
 - delay(y)** ▶ suspende o processo por y milisegundos
 - difftime(x,y)** ▶ diferença em segundos entre as horas x e y no formato interno do sistema operacional
 - Struct tm** ▶ sec, min, hours, day, months, years, ...

Copyright © 2012 Accenture All rights reserved.

82

Funções de Tempo

Laboratório

- Compilar os programas do diretório `/curso_qnx/timers`
 - delay – implementação de espera pelo comando `delay`
 - sleep – implementação de espera pelo comando `sleep`
 - Interval2 – exemplo de medição de intervalo de tempo com alta precisão
- Implementar um programa para pegar a data no formato interno do QNX e exibir no formato `dd/mm/aaaa - hh:mm:ss`
- Implementar um programa para receber a data e hora no formato `dd/mm/aaaa - hh:mm:ss` e fique aguardando essa hora ocorrer para encerrar sua execução

Copyright © 2012 Accenture All rights reserved.

83