TRABALHANDO COM ARQUIVOS EM LOTE NO MS-DOS

Os arquivos .bat existem desde a época do “bom e velho” MS-DOS. No Windows 7 um arquivo em lotes pode ter a extensão .bat ou .cmd. Um arquivo .bat ou .cmd, ou arquivo em lotes, é um arquivo de texto, onde cada linha corresponde a um comando do sistema operacional. Quando executamos um arquivo .bat, os comandos são executados em seqüência. Qualquer comando que pode ser executado em uma janela do Prompt de comando (Iniciar -> Todos os programas -> Acessórios -> Prompt de comando) poderá fazer parte de um arquivo .bat.

Um dos usos mais comuns para arquivos .bat é para a criação de scripts de logon, nos quais são colocados comandos que executam uma série de operações de configuração do ambiente do usuário, como a montagem de drives de rede, acesso a impressoras de rede, definição de variáveis de ambiente e assim por diante. Devido à facilidade e simplicidade de criação são de grande ajuda para automatizar tarefas repetitivas. A seguir iremos estudar com mais detalhes os comandos utilizados em arquivos .bat e .cmd.

Com os arquivos em lotes, também denominados programas em lotes ou scripts, é possível simplificar tarefas rotineiras ou repetitivas. Um arquivo em lotes é um arquivo de texto não formatado que contém um ou mais comandos e tem extensão de nome de arquivo .bat ou .cmd. Quando o nome do arquivo é digitado no prompt de comando, o ambiente de execução Cmd.exe executa os comandos do respectivo arquivo, em seqüência conforme são exibidos no arquivo (ou seja, da primeira até a última linha, em seqüência).

Qualquer comando pode ser incluído em um arquivo em lotes, incluindo todos os comandos internos, tais como dir, copy, etc. Alguns comandos como, por exemplo, for, goto e if permitem o processamento condicional dos comandos no arquivo em lotes. Por exemplo, o comando if executa um comando com base nos resultados de uma condição. Outros comandos permitem o controle de entradas e saídas e a chamada de outros arquivos em lotes.

Os códigos de erro padrão retornados pela maior parte dos aplicativos são 0 caso não tenham ocorrido erros e 1 (ou um valor superior) caso tenham ocorrido erros. Consulte a documentação de ajuda de seu aplicativo para determinar o significado dos códigos de erro específicos.

A seguir apresentamos os principais comandos para arquivos .bat, bem como exemplos de utilização.

Comando Call:

Este comando chama um programa em lotes a partir de outro, sem interromper o programa de onde foi feita a chamada. O comando call aceita rótulos como o destino da chamada.

Nota: O comando call não tem efeito algum na linha de comando quando usado fora de um script ou arquivo em lotes.

Sintaxe para o comando call:

call [drive:][caminho] nome\_do\_arquivo [parâmetros]

call :rótulo [argumentos]

Onde temos os seguintes parâmetros:

[drive:][caminho] nome\_do\_arquivo: Este parâmetro define o nome do arquivo que será chamado. O nome do arquivo deve ter a extensão .bat ou .cmd.  
Exemplo: CALL D:scriptsrotinas.bat  
parâmetros: Especifica qualquer informação de linha de comando exigida pelo programa em lotes, incluindo opções de linha de comando, nomes de arquivo, parâmetros em lotes (isto é, %0 a %9) ou variáveis (por exemplo, %taxa\_de\_transmissão%). Podemos passar até dez argumentos na linha de comando, quando chamamos um arquivo .bat ou .cmd. Estes parâmetros são associados às variáveis %0, %1 até %9. Utilizamos estas variáveis, dentro do arquivo .bat, para acessar o valor dos parâmetros que foram passados na linha de comando.  
:rótulo: Especifica o rótulo para o qual o controle do programa em lotes deverá saltar. Em um arquivo em lotes podemos criar rótulos, os quais identificam seções do arquivo. Podemos utilizar rótulos para iniciar a execução a partir de um determinado ponto do arquivo. O uso do comando call com esse parâmetro criará um novo contexto de arquivo em lotes e passará o controle para a instrução posterior ao rótulo especificado. Na primeira vez que o final do arquivo em lotes for encontrado (isto é, após o controle saltar para o rótulo), o controle retornará para a instrução após a instrução call. Na segunda vez que o final do arquivo em lotes for encontrado, o script em lotes será finalizado.  
/?: Exibe informações de ajuda no prompt de comando. Por exemplo, abra uma janela de Prompt de comando, digite call /? e pressione Enter para obter informações sobre o comando call.  
Alguns exemplos:

Vamos supor que a partir do arquivo script1.bat você precisa chamar o arquivo script2.bat, que está na pasta C:Administraçãoscripts. Para isso você pode usar o seguinte comando:

call C:Administraçãoscriptsscript2.bat

Agora vamos supor que, além de chamar o arquivo script2.bat, você precisa passar três parâmetros para este arquivo. Neste caso você pode utilizar o seguinte comando:

call C:Administraçãoscriptsscript2.bat %1 %2 %3

O arquivo script2.bat recebe os parâmetros, é executado e após ser finalizado a execução continua no arquivo script1.bat, a partir da primeira linha após a linha com o comando call.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Novidade do Windows Vista e do Windows 7: A expansão de referências de argumento de script em lotes (%0, %1,…, %9) foi bastante melhorada, conforme descrevo a seguir.

%\* em um script em lotes refere-se a todos os argumentos (ex.: %1 %2 %3 %4 %5 …)  
A substituição de parâmetros em lotes (%n) foi aprimorada. Agora é possível usar a seguinte sintaxe opcional:

%~1 – expande %1 removendo quaisquer aspas (“)

%~f1 – expande %1 para um nome de caminho totalmente qualificado

%~d1 – expande %1 para somente uma letra de unidade

%~p1 – expande %1 para somente um caminho

%~n1 – expande %1 para somente um nome de arquivo

%~x1 – expande %1 para somente uma extensão de arquivo

%~s1 – o caminho expandido contém somente nomes curtos

%~a1 – expande %1 para atributos de arquivo

%~t1 – expande %1 para data/hora do arquivo

%~z1 – expande %1 para o tamanho do arquivo

%~$PATH:1 – pesquisa as pastas listadas na variável de ambiente PATH e expande %1 para o nome totalmente qualificado da primeira encontrada. Se nome da variável de ambiente não estiver definido e o arquivo não for encontrado pela pesquisa, esse modificador será expandido para a cadeia de caracteres vazia

Os modificadores podem ser combinados para obter resultados compostos:

%~dp1 – expande %1 somente para uma letra da unidade e caminho

%~nx1 – expande %1 somente para uma extensão e nome de arquivo

%~dp$PATH:1 – pesquisa as pastas listadas na variável de ambiente PATH para %1 e expande para a letra da unidade e caminho da primeira encontrada.

%~ftza1 – expande %1 para uma linha de saída do tipo DIR

Nos exemplos acima %1 e PATH podem ser substituídos por outros valores válidos. A sintaxe %~ é finalizada por um número de argumento válido. Os modificadores %~ não podem ser usados com %\*

O Comando Echo:

Este comando é utilizado para ativar/desativar o recurso “eco” de comandos ou para exibir uma mensagem. Quando utilizado sem parâmetros, echo exibe a configuração de eco atual, isto é, se está ativado ou desativado.

Sintaxe:

echo [{on|off}] [mensagem]

Parâmetros:

{on|off}: Especifica se o recurso de eco de comandos deve ser ativado ou desativado. On ativa e Off desativa o recurso. Ativar o recurso significa que os comandos serão exibidos à medida que são executados, ou seja, o próprio texto dos comandos será exibido e não apenas os resultados da execução de cada comando.  
mensagem: Especifica o texto a ser exibido na tela.  
/?: Exibe informações de ajuda no prompt de comando. Por exemplo, abra uma janela de Prompt de comando, digite echo /? e pressione Enter para obter informações sobre o comando echo.  
O comando echo mensagem é útil quando o eco está desativado. Para exibir uma mensagem com várias linhas, sem exibir outros comandos, você pode incluir vários comandos echo mensagem depois do comando echo off no arquivo em lotes.

Se você utilizar echo off, o prompt de comando não será exibido na tela, enquanto o arquivo estiver sendo executado. Para exibir o prompt de comando, digite echo on. Para impedir o eco de uma linha, insira um sinal de arroba (@) na frente do comando.

Para fazer o eco de uma linha em branco na tela, digite simplesmente:

echo

Para exibir um pipe (|) ou um caractere de redirecionamento (< ou >) ao utilizar o comando echo, use um circunflexo imediatamente antes do pipe ou caractere de redirecionamento (por exemplo, ^>, ^< ou ^| ). Se precisar usar o próprio caractere circunflexo (^), use dois (^^). Os caracteres de redirecionamento são especialmente úteis para redirecionar a saída de um comando, para um arquivo de texto ou para a impressora. Por exemplo, o comando a seguir, executado na pasta-raiz do disco rígido, irá listar todas as pastas e todos os arquivos do disco rígido e a listagem será direcionada para um arquivo chamado conteúdo.txt. dir/s > conteúdo.txt

O sinal de > informa ao Windows 7 (na verdade este comando funciona desde a época do MS-DOS) que a listagem de saída, ao invés de ser exibida no vídeo, deve ser redirecionada para o arquivo chamado conteúdo.txt, na pasta atual.

Vejamos alguns exemplos:

O exemplo a seguir é um arquivo em lotes que inclui algumas linhas de mensagem. Observe que o eco não é desabilitado, pois não temos um comando echo off no início do arquivo.

@echo.

@echo Bem vindo: %username%

@echo Data atual:

@Date/t

@echo Hora atual:

@Time/t

@echo.

Ao executarmos este arquivo obteremos o seguinte resultado (supondo que o usuário logado é o usuário juliob):

Bem vindo: juliob

Data atual:

12/01/2012

Hora atual:

17:24

ECHO está ativado.

Alguns comentários sobre o exemplo anterior:

Utilizamos o sinal @, no início de cada linha, para evitar que o comando fosse exibido durante a execução.

Observe a segunda linha do exemplo:

@echo Bem vindo: %username%

Nesta linha utilizamos a variável de ambiente username. Para exibir o valor de uma variável de ambiente, devemos colocar o nome da variável entre sinais de porcentagem – %. Ao executar o comando, o Windows 7 substitui o nome da variável pelo seu valor. O Windows 7 utiliza uma série de variáveis de ambiente, para controlar algumas configurações do Sistema Operacional.

Exemplo prático: Para obter uma lista das variáveis de ambiente disponíveis, siga os passos indicados a seguir:

1. Abra um Prompt de comando: Iniciar -> Todos os programas -> Acessórios -> Prompt de comando.

2. Digite o comando set e pressione Enter. Na listagem a seguir temos um exemplo do resultado da execução do comando set:

ALLUSERSPROFILE=C:ProgramData

APPDATA=C:UsersjuliobAppDataRoaming

CommonProgramFiles=C:Program FilesCommon Files

COMPUTERNAME=MICROXP01

ComSpec=C:Windowssystem32cmd.exe

FP\_NO\_HOST\_CHECK=NO

HOMEDRIVE=C:

HOMEPATH=Usersjuliob

LOCALAPPDATA=C:UsersjuliobAppDataLocal

LOGONSERVER=\MICROXP01

NUMBER\_OF\_PROCESSORS=2

OS=Windows\_NT

Path=C:Windowssystem32;C:Windows;C:WindowsSystem32Wbem

PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

PROCESSOR\_ARCHITECTURE=x86

PROCESSOR\_IDENTIFIER=x86 Family 15 Model 4 Stepping 7, GenuineIntel

PROCESSOR\_LEVEL=15

PROCESSOR\_REVISION=0407

ProgramData=C:ProgramData

ProgramFiles=C:Program Files

PROMPT=$P$G

PUBLIC=C:UsersPublic

SystemDrive=C:

SystemRoot=C:Windows

TEMP=C:UsersjuliobAppDataLocalTemp

TMP=C:UsersjuliobAppDataLocalTemp

USERDOMAIN=MICROXP01

USERNAME=juliob

USERPROFILE=C:Usersjuliob

windir=C:Windows

Qualquer uma destas variáveis pode ser utilizada em arquivos .bat ou .cmd, bastando usar o nome da variável entre sinais de porcentagem (%), conforme já citado anteriormente.

Para desativar o comando echo e não repetir o próprio comando echo, inclua um sinal de arroba (@) antes do comando, desta maneira:

@echo off

Observe que, no nosso exemplo, utilizamos o sinal de @ antes de todos os comandos, para que estes não fossem exibidos durante a execução do arquivo.

Os Comandos Setlocal/Endlocal:

Este comando é utilizado para a localização de variáveis de ambiente em um arquivo em lotes. A localização continuará até que um comando Endlocal correspondente seja encontrado ou o final do arquivo em lotes seja alcançado. A principal utilização deste comando é para alterar variáveis de ambiente, durante a execução do arquivo em lotes. Também é possível criar uma variável de ambiente utilizando o comando Setlocal. Esta variável continuará existindo, até que um comando Endlocal seja encontrado.

Nota: Se utilizarmos Setlocal fora de um arquivo em lotes ou de um script, o comando não tem nenhum efeito.

Como alterar variáveis de ambiente, usando Setlocal:

Podemos utilizar o comando Setlocal para alterar o valor de uma ou mais variáveis de ambiente, durante a execução do arquivo .bat ou .cmd. As alterações de ambiente feitas após a execução do comando Setlocal são locais para o arquivo em lotes, isto é, somente são válidas no contexto de execução do arquivo. O interpretador de comandos Cmd.exe restaura configurações anteriores quando encontra um comando Endlocal ou alcança o final do arquivo em lotes.

É possível ter mais de um comando Setlocal ou Endlocal em um programa em lotes (isto é, comandos aninhados).

Exemplo: Vamos supor que você precisa executar um programa que precisa que a pasta temporária seja a pasta C:Temp. No Windows 7 a pasta temporária é personalizada para cada usuário e fica no seguinte caminho: C:UsersNome\_do\_usuárioAppDataLocalTemp. Neste caso podemos usar um arquivo .bat para alterar a localização da pasta temporária para C:Temp, para chamar o respectivo programa. Ao encerrar o programa, o arquivo .bat também será encerrado com um chamado do comando Endlocal, o que faz com que a variável de ambiente TEMP, a qual define a localização da pasta temporária, seja restaurada ao seu valor original. Na listagem a seguir temos o exemplo proposto, onde estamos executando o aplicativo C:Jogosdn.exe:

rem \*\*\*\*\*\*\* Altera a pasta temp, para a execução de dn.exe \*\*\*

>@echo off

@setlocal

@temp=C:temp

@C:jogosdn.exe

@endlocal

Nota: Para que este exemplo funcione corretamente, a pasta C:Temp já deve existir. Mais adiante aprenderemos comandos mais sofisticados que permitem que façamos um teste: Se a pasta C:Temp já existe, seguimos com a execução do arquivo, mas caso contrário, criamos a pasta.

Observe a utilização de Endlocal, para forçar a restauração da variável de ambiente temp, ao seu valor original.

O Comando Rem

O comando Rem é utilizado para a inclusão de comentários em arquivos em lote. Também pode ser utilizado para inserir linhas em branco, durante a execução do programa, para isto bastando incluir um comando Rem sem nenhum comentário.

Nota: Não é possível usar um caractere de redirecionamento “(“ ou “)” nem de pipe (|) em um comentário do arquivo em lotes.

No exemplo dos comandos Setlocal/Endlocal, utilizamos o comando Rem para inserir um comentário no início do arquivo.

O Comando if

O comando if é utilizado para a execução de testes lógicos, do tipo Verdadeiro ou Falso. Com base no resultado do teste, um ou outro comando (ou conjunto de comandos) será executado. Considere o exemplo a seguir:

echo off

format a: /s

if not errorlevel 1 goto end

echo Ocorreram erros durante a formatação.

end

echo Final do programa em lote.

Neste exemplo utilizamos o comando format para formatar o disquete e transferir o Sistema operacional: opção /s. Na seqüência testamos para verificar se houve algum erro com a formatação:

if not errorlevel 1 goto end

Esta linha testa se o código de erro é diferente de um, e caso seja, o comando goto é utilizado para desviar a execução para a linha indicada pelo rótulo end. Caso o código de erro seja 1, o teste será falso e a mensagem “Ocorreram erros durante a formatação”, será exibida.

Se o teste é verdadeiro, o comando que segue o teste será executado. Se o teste for falso, o comando da próxima linha será executado ou os comandos da cláusula else, caso esta exista. Considere os seguintes exemplos genéricos de utilização do comando if, onde utilizamos a cláusula else:

if [not] errorlevel número comando [else expressão]

if [not] string1==string2 comando [else expressão]

if [not] exist arquivo comando [else expressão]

Após esta breve apresentação do comando If..else, vamos estudá-lo em detalhes.

Parâmetros e exemplos para o comando If..else.

not: É o operador lógico Not. Este operador inverte o valor de um teste lógico, ou seja, se o teste for Verdadeiro, Not Teste é igual a falso; se o teste for Falso, Not Teste é igual a verdadeiro. Quando utilizamos o Not com o teste, o comando que segue o If somente será executado, quando o teste for Falso, pois Not Falso retornará Verdadeiro e o comando será executado.  
errorlevel número: Retorna verdadeiro apenas se o último comando executado retorna um código de erro igual ou maior do que o especificado. Cada comando tem um conjunto de códigos de erro que são retornados.  
command: Define o comando que será executado, caso a condição do teste seja verdadeiro. Pode ser qualquer comando reconhecido pelo Windows 7.  
string1==string2: Retorna verdadeiro apenas se as duas variáveis contiverem o mesmo valor. Os valores podem ser valores definidos no próprio programa ou valores passados como parâmetros, na execução do arquivo .bat, utilizando os indicadores %1, %2 e assim por diante.  
exist arquivo: Utilizamos exist para verificar se um arquivo existe. Se o arquivo existir, será retornado Verdadeiro; caso contrário será retornado falso.  
Para efetuar os testes do comando If, nós utilizamos os seguintes operadores de comparação

**Operador Descrição**

QUE igual à

NEQ diferente

LSS menor do que

LEQ menor ou igual à

GTR maior do que

GEQ maior ou igual à

/i: Utilizamos esta opção para forçar uma comparação entre duas variáveis de texto, de tal maneira que sejam ignoradas maiúsculas ou minúsculas. Esta cláusula pode ser utilizada diretamente em um teste do tipo texto1=texto2.  
defined variável de ambiente: Esta opção é semelhante à opção exist, porém é utilizada para verificar se uma determinada variável de ambiente existe e está definida, ou seja, possui um valor associado à variável. Ao utilizarmos defined variável, as três seguintes variáveis serão adicionadas: %errorlevel%, %cmdcmdline% e %cmdextversion%.  
%errorlevel%: expande uma representação de seqüência do valor atual de errorlevel, desde que não exista ainda uma variável de ambiente com o nome ERRORLEVEL; nesse caso, você obterá o valor dessa variável. O exemplo a seguir ilustra como você pode usar errorlevel depois de executar um programa em lotes:  
goto resposta%errorlevel%

:resposta0

:resposta1

echo O programa retornou o código 1

goto end

:end

echo concluído!

O programa recebe um valor retornado por %errorlevel%, e este valor é anexado (isto significa expande) com resposta. Por exemplo, se %errorlevel% tiver o valor 1, o resultado final do comando fica:

goto resposta1

Você também pode usar os operadores de comparação

if %errorlevel% LEQ 1 goto okay

%cmdcmdline%: expande a linha do comando original passada para Cmd.exe antes de qualquer processamento pelo Cmd.exe, desde que não exista ainda uma variável de ambiente com o nome cmdcmdline; nesse caso, você obterá o valor de cmdcmdline.  
% cmdextversion %: expande uma representação de seqüência do valor atual de cmdextversion, desde que não exista ainda uma variável de ambiente com o nome CMDEXTVERSION; nesse caso, você obterá o valor de CMDEXTVERSION.  
expressão: Especifica um comando e quaisquer parâmetros de linha de comando a ser passado ao comando em uma cláusula else.  
/?: Exibe ajuda no prompt de comando.  
Vamos apresentar alguns exemplos práticos para utilização do comando If..else.

Exemplo Prático 01: Utilizando If para verificar se um determinado arquivo existe. Caso o arquivo não exista, a mensagem “ARQUIVO NÃO EXISTE”, será emitida; caso exista, a mensagem “O ARQUIVO EXISTE” será emitida.

Para resolver o problema proposto, considere a listagem a seguir:

@echo.

@echo Verificando se C:dados.txt existe

@If not exist C:dados.txt. (

@echo “O ARQUIVO NÃO EXISTE”.

else (

@echo “O ARQUIVO EXISTE”.

)

Observe que o uso do comando If, com uma cláusula Else, precisa obedecer uma sintaxe bem específica, conforme indicado a seguir:

IF NOT EXIST nome\_do\_arquivo. (

comando.

ELSE (

>comando.

)

O Comando FOR

Utilizamos o comando For para executar um determinado comando para cada arquivo em um conjunto de arquivos. Considere o exemplo a seguir:

Para exibir o conteúdo de todos os arquivos da pasta atual que tenham a extensão .bat ou .txt usando a variável substituível %f, utilize o seguinte comando:

for %%f in (\*.bat \*.txt) do type %f

Para excluir todos os arquivos da pasta atual que tenham a extensão .tmp ou .wks usando a variável substituível %f, utilize o seguinte comando:

for %%f in (\*.tmp \*.wks) do del %f

Quando utilizamos o comando for, o primeiro valor do conjunto definido na cláusula in, substitui %variável ou %%variável e, então, o comando especificado processa esse valor. Esse procedimento continua até que todos os arquivos (ou grupos de arquivos) que correspondam ao valor do conjunto (cláusula in) sejam processados.

Vamos ver alguns detalhes sobre o comando for.

Sintaxe, segundo a ajuda do Windows 7:

for {%variável|%%variável} in (conjunto) do comando [opções\_da\_linha\_de\_comando]

Vamos detalhar os parâmetros deste comando:

{%variável|%%variável}: Este parâmetro é obrigatório. Representa um parâmetro substituível. Utilize %variável para executar o comando for a partir do prompt de comando. Utilize %%variável para executar o comando for dentro de um arquivo em lotes. As variáveis diferenciam maiúsculas de minúsculas e devem ser representadas com um valor alfabético, como %%A, %%B ou %%C. Nos exemplos do início deste tópico, utilizamos %%f.  
(conjunto): Também é obrigatório. Especifica um ou mais arquivos, pastas, intervalo de valores ou seqüências de texto a serem processados com o comando especificado. Os parênteses são obrigatórios. Nos primeiros exemplos utilizamos (\*.bat \*.txt) e no segundo (\*.tmp \*.wks).  
comando: Obrigatório. Especifica o comando a ser executado em cada arquivo, pasta, intervalo de valores ou seqüência de texto incluído no (conjunto) especificado.  
opções\_da\_linha\_de\_comando: Especifica quaisquer opções da linha de comando a serem utilizadas com o comando especificado.  
/?: Exibe ajuda no prompt de comando.  
Nota: Para evitar confusão com os parâmetros em lote de %0 a %9, qualquer caractere pode ser utilizado para variável, exceto os algarismos de 0 a 9. Para arquivos em lote simples, basta um único caractere, como %%f que foi utilizado nos exemplos do início deste tópico.

Alguns exemplos sobre a especificação do conjunto de arquivos sobre o qual agirá o comando for:

O parâmetro conjunto pode representar um único grupo de arquivos ou vários grupos de arquivos. Podem ser utilizados os caracteres curingas: \* e ? para especificar um conjunto de arquivos. São válidos os seguintes conjuntos de arquivos:

(\*.tmp) -> Todos os arquivos com a extensão .tmp, na pasta atual.  
(\*.doc \*.xls \*.mdb) -> Todos os arquivos com a extensão .doc, .xls ou .mdb, na pasta atual.  
(a\*.doc a\*.rpt b\*.doc b\*.rpt) -> Todos os arquivos com o nome iniciando por a, com as extensões .doc ou .rpt e todos os arquivos com o nome iniciando por b, com as extensões .doc ou .rpt, na pasta atual.

O Comando Pause

O comando Pause é utilizado para suspender o processamento de um arquivo em lotes e exibe uma mensagem solicitando ao usuário para pressionar qualquer tecla para continuar.

Sintaxe:

pause

Parâmetros:

/?: Exibe ajuda no prompt de comando.  
Quando o comando Pause é executado, a seguinte mensagem será exibida:

Pressione qualquer tecla para continuar. . .

Quando o usuário pressiona uma tecla, a execução continua com o comando após o Pause.

Se você pressionar CTRL+C para interromper um programa em lotes, é exibida a seguinte mensagem:

Finalizar trabalho em lotes (S/N)?

Se você pressionar S (para sim) em resposta a essa mensagem, o programa em lotes será finalizado e o controle retornará ao sistema operacional. Portanto, você pode inserir o comando pause antes de uma seção do arquivo em lotes, seção este que você pode não querer executar. Enquanto pause suspende o processamento do programa em lotes, você pode pressionar CTRL+C e, em seguida, S para interromper o programa.

A seguir coloco um exemplo que está disponível na Documentação do Windows 7 e que salienta bem o uso do comando Pause.

Para criar um programa em lotes que emita uma mensagem solicitando ao usuário para trocar o disco em uma das unidades, digite:

@echo off

Nesse exemplo, todos os arquivos do disco na unidade A são copiados para a pasta atual. Depois que a mensagem é exibida solicitando a inserção de outro disco na unidade A, o comando pause suspende o processamento para que você troque os discos e, em seguida, pressione qualquer tecla para continuar o processamento. Este programa em lotes, especificamente, executa um loop sem fim. O comando goto BEGIN envia o interpretador de comandos para o início do arquivo em lotes, onde foi definido o rótulo begin. Para finalizar este arquivo em lotes, pressione CTRL+C e, em seguida, S.

O Comando Shift

O comando Shift é utilizado para alterar a posição de parâmetros em lotes de um arquivo em lotes. Os parâmetros são passados na linha de comando e são acessados dentro do arquivo de lote, utilizando-se %0, %1, %2 e assim por diante. Por exemplo, para executar o arquivo dados.bat, passando dois parâmetros, utilize a seguinte sintaxe:

dados.bat C:teste.txt out

Sintaxe para o comando Shift:

shift

Parâmetros: nenhum:

O comando shift altera os valores dos parâmetros em lotes de %0 a %9 copiando cada parâmetro para o anterior. Em outras palavras, o valor de %1 é copiado para %0, o valor de %2 é copiado para %1, e assim por diante. Isso é útil para gravar um arquivo em lotes que executa a mesma operação em qualquer número de parâmetros.

Também é possível usar o comando shift para criar um arquivo em lotes que aceite mais de dez parâmetros em lotes. Se você especificar mais de dez parâmetros na linha de comando, aqueles que forem exibidos depois do décimo (%9) serão alternados, um por vez, para %9.

Nota: Não existe um comando shift regressivo. Após executar o comando shift, você não poderá recuperar o primeiro parâmetro em lotes (%0) que existia antes da alternância.

É importante ressaltar que além dos comandos específicos para arquivos em lotes: Call, Echo, Endlocal, For, Goto, If, Pause, Rem, Setlocal e Shift, é possível utilizar qualquer comando do Windows 7 em um arquivo em lotes. Por exemplo, você pode utilizar o comando net e todas as suas opções em um arquivo em lotes, para montar drivers, acertar a hora do sistema com um Controlador de Domínio da rede, etc. Qualquer comando válido para o Windows 7, pode ser utilizado em um arquivo em lotes.