Windows 2000 - zbiory wsadowe

Zbiory wsadowe (*ang. batch*) są wykorzystywane w celu uproszczenia pewnych powtarzających się operacji, wymagających każdorazowo podania jednego (zwykle długiego), lub też wielu poleceń dostępnych w trybie interakcyjnym. Oprócz takich poleceń można w nich stosować również pewne konstrukcje przeznaczone specjalnie do tego celu. Konstrukcje te określają składnię języka wsadowego i są zależna od interpretera zleceń. W systemie Windows 2000 domyślnym interpreterem poleceń jest program *CMD.EXE,* który wywodzi się wprost z ubogiego – choć ewoluującego w kolejnych wersjach systemu – interpretera *command.com.*

Zbiór wsadowy to nic innego jak zbiór tekstowy (ASCII), z rozszerzeniem *.bat*, lub *.cmd* zawierający dowolny ciąg zewnętrznych i wewnętrznych poleceń systemu. Wykonanie pliku wsadowego wymaga podania jedynie jego nazwy (bez rozszerzenia) z ewentualnymi parametrami. Wykonywanie pliku wsadowego można przerwać przez Ctrl+C. Trzeba tu dodać, że choć zbiór dostępnych poleceń jest dosyć ubogi, to jednak wykorzystując doświadczenie, a także korzystając ze zbiorów pomocniczych oraz pewnych trików można tworzyć zaawansowane zbiory wsadowe. Niżej omówiono niektóre elementy niezbędne do tworzenia zbiorów wsadowych.

1. Wprowadzanie poleceń

Nazwy poleceń wprowadza się bez rozróżnienia małych i dużych liter, opcje poleceń w większości nie są wrażliwe na wielkość liter. Polecenie może być kontynuowane w drugim (kolejnym) wierszu dzięki umieszczeniu znaku daszka (^) na końcu poprzedzającego wiersza. Znak daszka jest także znakiem odwołania- zapobiega on przetwarzaniu prze interpreter znaku następującego po nim. Na przykład chcąc wyprowadzić na ekran opis używania programu wsadowego możemy napisać:

```
echo Sposób wywołania: %0 ^<nazwa_pliku^>
```

Polecenia mogą być łączone za pomocą znaku ampersand "&" – wykonywane wtedy są sekwencyjnie niezależnie od wyniku zakończenia poprzedniego polecenia. Np:

```
if /i {%1}=={/?} (call :HELP %2) & (goto :HELPEXIT)
```

Polecenia mogą być także wykonywane warunkowo w zależności od powodzenia polecenia poprzedzającego. W tym celu należy polecenia połączyć znakami && lub ||. Odpowiednio:

polecenie1 && *polecenie2* – *polecenie2* zostanie wykonane jeżeli *polecenie1* zakończy się sukcesem
polecenie1 || *polecenie2* - *polecenie2* zostanie wykonane jeżeli *polecenie1* zakończy się niepowodzeniem

Przykład: md %katalog% && cd %katalog% - jeżeli utworzenie katalogu powiedzie się przejdziemy do niego.

Przekierowanie danych wejściowych i wyjściowych polecenia, możemy dokonywać za pomocą znaków >, >>, < oraz |. Każde polecenie posiada trzy strumienie danych: wejściowy (0), wyjściowy (1) i błędów (2). Przykłady:

polecenie <*plik* – pobranie z pliku standardowego wejścia
polecenie >*plik*,
polecenie 1>*plik* – wysłanie standardowego wyjścia do pliku
polecenie >>*plik*,

polecenie 1>>*plik* – dołączenie standardowego wyjścia do pliku
polecenie 2>*plik* – wysłanie komunikatu o błędzie do pliku
polecenie 2>>*plik* – dołączenie komunikatu o błędzie do pliku
polecenie >*plik* 2>&1 – połączenie strumienia wyjściowego i błędów i wysłanie do pliku
polecenie1| *polecenie2* – potok, łączy wyjście *polecenia1* ze standardowym wejściem *polecenia2*.

2. Wywoływanie interpretera:

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF] [[/S] [/C | /K] ciąg]

/C Uruchamia nowy interpreter i wykonuje polecenie określone przez ciąg i kończy działanie
/K Uruchamia nowy interpreter i wykonuje polecenie określone przez ciąg, ale pozostaje
/S Modyfikuje postępowanie z ciągiem po /C lub /K (zobacz niżej)
/Q Wyłącza echo
/D Wyłącza wykonywanie poleceń AutoRun z rejestru (zobacz niżej)
/A Powoduje, że dane przekazywane do potoku lub pliku są danymi ANSI
/U Powoduje, że dane przekazywane do potoku lub pliku są danymi Unicode
/T:fg Ustawia kolory tła i pierwszego planu (dodatkowe informacje uzyskasz wpisując COLOR /?)
/E:ON Włącza rozszerzenia poleceń (zobacz niżej)
/E:OFF Wyłącza rozszerzenia poleceń (zobacz niżej)
/F:ON Włącza znaki dokańczania nazw plików i katalogów (patrz poniżej)
/F:OFF Wyłącza znaki dokańczania nazw plików i katalogów (patrz poniżej)
/V:ON Włącza opóźnione rozwijanie zmiennych środowiskowych, traktując znak '!' jako ogranicznik.

Na przykład użycie /V:ON może spowodować, że napis !var! będzie rozwijać zmienną var w czasie wykonywania. Użycie składni var powoduje, że zmienne są rozwijane w czasie ich wprowadzania, co ma duże znaczenie w pętlach FOR.

/V:OFF Wyłącza opóźnione rozwijanie zmiennych środowiskowych.

Należy zauważyć, że napis złożony z kilku poleceń oddzielonych separatorem poleceń '&&' jest akceptowalny dla ciągów, jeżeli jest on ujęty w cudzysłowy. Również, ze względu na zachowanie zgodności, przełącznik /X ma takie samo działanie jak /E:ON, /Y działa tak samo jak /E:OFF, a /R działa tak samo jak /C. Inne przełączniki są ignorowane.

W przypadku użycia przełączników /C lub /K, pozostała część wiersza polecenia po przełączniku jest przetwarzana jako wiersz polecenia, który obowiązują następujące reguły przetwarzania znaków cudzysłowu (''):
1. Znaki cudzysłowu w wierszu polecenia są zachowywane, jeżeli spełnione są wszystkie poniższe warunki:
   - nie użyto przełącznika /S,
   - użyto dokładnie dwóch znaków cudzysłowu,
   - pomiędzy znakami cudzysłowu nie występują żadne znaki specjalne, takie jak: `&<>()@^|`,
   - pomiędzy dwoma znakami cudzysłowu znajduje się co najmniej jeden znak odstępu,
   - ciąg znajdujący się pomiędzy dwoma znakami cudzysłowu jest nazwą pliku wykonywalnego.
2. W pozostałych przypadkach, tradycyjne działanie polega na sprawdzeniu, czy pierwszym znakiem jest znak cudzysłowu i, jeżeli tak, usunięcie tego znaku oraz usunięcie ostatniego znalezionego znaku cudzysłowu w wierszu polecenia, z zachowaniem całego tekstu, znajdującego się za ostatnim znakiem cudzysłowu.

Jeśli nie podano opcji /D w wierszu polecenia, a następnie uruchomiono CMD.EXE, wyszukiwane będą poniższe zmienne rejestru REG_SZ/REG_EXPAND_SZ i jeśli jedna z nich lub obie są obecne, wykonywane są jako pierwsze.

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun
i/lub
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\AutoRun

Rozszerzenia poleceń są włączone domyślnie. Rozszerzenia dla poszczególnych wywołań można wyłączyć przy użyciu przełącznika /E:OFF. Można włączać lub wyłączać rozszerzenia dla wszystkich wywołań programu CMD.EXE na komputerze i/lub sesji logowań użytkownika przez ustawienie w rejestrze jednej lub obu następujących wartości REG_DWORD za pomocą programu REGEDT32.EXE:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\EnableExtensions
i/lub
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\EnableExtensions

na 0x1 albo 0x0. Ustawienie określone przez użytkownika ma pierwszeństwo przed ustawieniem komputera. Przełączniki wiersza polecenia mają pierwszeństwo przed ustawieniami rejestru.

Rozszerzenia poleceń obejmują zmiany i/lub dodatki dla następujących poleceń:
DEL lub ERASE, COLOR, CD lub CHDIR, MD lub MKDIR, PROMPT, PUSHD, POPD, SET, SETLOCAL, ENDLOCAL, IF, FOR, CALL, SHIFT.
Aby uzyskać szczegółowe informacje na temat danego polecenia, wpisz *nazwa_polecenia /?*.

Opóźnione rozwijanie zmiennych środowiskowych NIE jest włączone domyślnie. Opóźnione rozwijanie zmiennych środowiskowych dla poszczególnych wywołań programu CMD.EXE można włączać lub wyłączać przy użyciu przełącznika /V:ON lub /V:OFF. Można włączać lub wyłączać zakończenia dla wszystkich wywołań programu CMD.EXE na komputerze i/lub sesji logowań użytkownika przez ustawienie w rejestrze jednej lub obu następujących wartości REG_DWORD za pomocą programu REGEDT32.EXE:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\DelayedExpansion
i/lub
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\DelayedExpansion

na 0x1 albo 0x0. Ustawienie określone przez użytkownika ma pierwszeństwo przed ustawieniem komputera. Przełączniki wiersza polecenia mają pierwszeństwo przed ustawieniami rejestru. Jeśli opóźnione rozwijanie zmiennych środowiskowych jest włączone, wtedy znak wykrzyknika może być używany do podstawiania wartości zmiennej środowiskowej w czasie wykonywania. Dokańczanie nazw plików i katalogów NIE jest włączone domyślnie. Można włączać lub wyłączać dokańczanie nazw plików dla poszczególnych wywołań programu CMD.EXE przy użyciu przełącznika /F:ON lub /F:OFF. Można włączać lub wyłączać dokańczanie dla wszystkich wywołań programu CMD.EXE na komputerze i/lub sesji logowań użytkownika przez ustawienie w rejestrze jednej lub obu następujących wartości REG_DWORD za pomocą programu REGEDT32.EXE:

HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\CompletionChar
HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\PathCompletionChar
i/lub

HKEY_CURRENT_USER\Software\Microsoft\Command Processor\CompletionChar
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\PathCompletionChar

na wartość szesnastkową znaku kontrolnego dla poszczególnej funkcji (np. 0x4 jest Ctrl-D a 0x6 jest Ctrl-F). Ustawienie określone przez użytkownika ma pierwszeństwo przed ustawieniem komputera. Przełączniki wiersza polecenia mają pierwszeństwo przed ustawieniami rejestru. Jeśli dokańczanie jest włączone za pomocą przełącznika /F:ON, dwoma znakami kontrolnymi są: Ctrl-D dla dokańczania nazw katalogów i Ctrl-F dla dokańczania nazw plików. Aby wyłączyć poszczególne znaki dokańczania w rejestrze, użyj wartości dla spacji (0x20), ponieważ nie jest to prawidłowy znak kontrolny.

Dokańczanie jest wywoływane po wpisaniu jednego z dwóch znaków kontrolnych. Funkcja dokańczania przenosi ciąg ścieżki na lewą stronę kursora, dołącza symbol wieloznaczny, jeśli go nie ma i buduje listę pasujących ścieżek. Następnie wyświetla pierwszą zgodną ścieżkę. Później, ponowne naciśnięcie tego samego znaku kontrolnego powtarza cykl przechodzenia przez listę pasujących ścieżek. Naciśnięcie klawisza Shift razem ze znakiem kontrolnym powoduje przechodzenie przez listę wstecz. Jeśli wiersz edytowany jest w dowolny sposób i naciśnięty zostanie ponownie znak kontrolny, zapisana lista pasujących ścieżek jest odrzucana i generowana jest nowa lista. To samo wystąpi w przypadku przełączenia między dokańczaniem nazw plików i katalogów. Jedyną różnicą między dwoma znakami kontrolnymi jest to, że znak dokańczania pliku dopasowuje zarówno nazwy plików, jak i katalogów, podczas gdy znak dokańczania katalogów dopasowuje jedynie nazwy katalogów. Jeśli dokańczanie jest używane dla wszystkich wbudowanych poleceń katalogów (CD, MD lub RD), to przyjmowane jest dokańczanie katalogów.

Kod dokańczania odpowiednio postępuje z nazwami plików, które zawierają spacje lub inne znaki specjalne, umieszczając cudzysłowy wokół pasującej ścieżki. W przypadku wycofania się i wywołania dokańczania wewnątrz wiersza, tekst z prawej strony kursora w punkcie, w którym wywoływano dokańczanie jest odrzucany.

3. Zmienne środowiska

Wartości zmiennych środowiska, tworzonych poleceniem SET [zmienna=[ciąg]], gdzie ciąg określa ciąg znaków, który ma być przypisany zmiennej, są dostępne wewnątrz zbiorów wsadowych po otoczeniu nazwy zmiennej znakami %, np. %zmienna% spowoduje zastąpienie jej odpowiednim ciągiem znaków.
Wpisz SET bez parametrów, aby wyświetlić bieżące zmienne środowiskowe. Przy włączonych rozszerzeniach poleceń polecenie SET zmienia się następująco:
1.Polecenie SET wywołane tylko z nazwą zmiennej, bez znaku równości lub wartości wyświetli wartość wszystkich zmiennych, których prefiks odpowiada nazwie podanej poleceniu SET. Na przykład:
SET P
- wyświetli wszystkie zmienne, których nazwy zaczynają się od litery 'P'
2. Polecenie SET nada zmiennej ERRORLEVEL wartość 1, jeśli nazwy tej zmiennej nie można znaleźć w bieżącym środowisku. Polecenie SET nie zezwala, aby znak równości (=) był częścią nazwy zmiennej. Polecenia SET ponadto posiada przełączniki:

SET /A wyrażenie
SET /P zmienna=[ciąg_monitu]

Przełącznik /A mówi, że ciąg po prawej stronie znaku równości jest wyrażeniem numerycznym, które jest obliczane. Program obsługujący obliczanie wyrażeń jest bardzo prosty i obsługuje następujące operacje (wymienione według malejącego pierwszeństwa):

() - grupowanie,
* / % - operatory arytmetyczne,

+ -	- operatory arytmetyczne,
<< >>	- przesunięcie logiczne,
&	- logiczne "i",
^	- logiczne wyłączne "lub",
|	- logiczne "lub",
= *= /= %= += -= ,	
&= ^= |= <<= >>=	- przypisanie
,	- separator wyrażeń

Jeśli używasz operatorów logicznych lub modulo, ujmij ciąg wyrażenia w cudzysłów. Wszelkie nie numeryczne ciągi w wyrażeniu są traktowane jako zmienne środowiskowe, których wartości, przed ich użyciem, muszą być konwertowane na liczby. Jeśli nazwa zmiennej środowiskowej jest podana, ale nie jest określona w bieżącym środowisku, to używana jest wartość zero. Pozwala to na działania arytmetyczne z użyciem zmiennych środowiskowych bez potrzeby wpisywania znaków %, by uzyskać ich wartości. Jeśli polecenie SET /A jest wykonywane z wiersza polecenia poza skryptem poleceń, to wyświetla ostateczną wartość wyrażenia. Operator przypisania wymaga nazwy zmiennej środowiska po swojej lewej stronie. Wartości liczbowe są liczbami dziesiętnymi, o ile nie są poprzedzone znakami 0x (wówczas są to liczby w postaci szesnastkowej), 0b (liczby dwójkowe) lub 0 (liczby ósemkowe). Zatem 0x12 jest tym samym co 0b10010 i tym samym co 022. Zwróć uwagę, że zapis ósemkowy może być mylący: 08 i 09 nie są poprawnymi liczbami, ponieważ 8 i 9 nie są prawidłowymi cyframi ósemkowymi.

Przełącznik /P zezwala na ustawienie wartości zmiennej dla wiersza wejścia wprowadzonego przez użytkownika. Wyświetla określony ciąg_monitu przed odczytaniem wiersza wejścia. Ciąg_monitu może być pusty.

Opóźnione rozwijanie zmiennych środowiskowych jest użyteczne do omijania ograniczeń bieżącego rozwinięcia, które mają miejsce gdy wiersz jest czytany, a nie kiedy jest wykonywany. Następujący przykład przedstawia problem z rozwinięciem zmiennej natychmiastowej:

```
set VAR=przed
if "%VAR%" == "przed" (
    set VAR=po
    if "%VAR%" == "po" @echo Jeśli to widzisz, to zadziałało
)
```

nigdy nie wyświetli komunikatu, ponieważ %VAR% w instrukcjach IF jest podstawiana, kiedy czytana jest pierwsza instrukcja IF, ponieważ logicznie zawiera główne polecenie IF, które jest instrukcją złożoną. Dlatego IF wewnątrz instrukcji złożonej faktycznie porównuje "przed" z "po", które nigdy nie będą sobie równe.
Opóźnione rozwijanie zmiennych środowiskowych zezwala na używanie innego znaku (wykrzyknik) do rozwijania zmiennych środowiskowych w czasie wykonywania. Jeśli opóźnione rozwijanie zmiennych jest włączone, powyższe przykład można zapisać jak poniżej, aby zadziałały zgodnie z oczekiwaniami:

```
set VAR=przed
if "%VAR%" == "przed" (
    set VAR=po
    if "!VAR!" == "po" @echo Jeśli to widzisz, to zadziałało
)
```

Przy włączonych rozszerzeniach poleceń, istnieje kilka dynamicznych zmiennych środowiskowych, które można rozwijać, ale które nie pojawiają się na liście zmiennych wyświetlanych przy użyciu polecenia SET. Te wartości zmiennych są obliczane dynamicznie za każdym razem, gdy wartość zmiennej jest

rozwijana. Jeśli użytkownik jawnie definiuje zmienną z jedną z tych nazw, wtedy definicja ta zastępuje nazwę dynamiczną opisaną poniżej:

%CD% - rozwija do ciągu katalogu bieżącego.
%DATE% - rozwija do bieżącej daty używając tego samego formatu co polecenie DATE.
%TIME% - rozwija do bieżącego czasu używając tego samego formatu co polecenie TIME.
%RANDOM% - rozwija do losowej liczby dziesiętnej między 0 a 32767.
%ERRORLEVEL% - rozwija do bieżącej wartości ERRORLEVEL.
%CMDEXTVERSION% - rozwija do bieżącej liczby wersji rozszerzeń procesora poleceń.
%CMDCMDLINE% - rozwija do oryginalnego wiersza polecenia, który wywołał procesora poleceń.

4. Parametry

Ponieważ podczas wywoływania zbioru wsadowego po jego nazwie możemy podać ciąg parametrów, konieczna jest możliwość odwoływania się do nich wewnątrz zbioru wsadowego. Jest to możliwe po użyciu symboli %0, %1, %2, … %9, przy czym %0 oznacza nazwę zbioru wsadowego, a dalsze parametry są oznaczane odpowiednio przez %1, %2, itd., np. po wywołaniu kopiuj.bat zbior.1 zbiór.2, wewnątrz zbioru kopiuj.bat symbol %0 oznacza nazwę zbioru wsadowego, %1 odpowiada nazwie zbior.1, a %2 nazwie zbior.2.
Ponadto zmienione zostały rozszerzenia odwołań do argumentów (%0, %1 itd.) pliku skryptu:
%* w skrypcie odnosi się do wszystkich argumentów (tzn. %1 %2 %3 %4 %5 ...)
Zostało ulepszone podstawianie parametrów pliku wsadowego (%n). Możesz teraz używać następującej opcjonalnej składni:

%~1 - rozwija %1 usuwając wszystkie obejmujące cudzysłowy (")
%~f1 - rozwija %1 do pełnej nazwy ścieżki
%~d1 - rozwija %1 tylko do litery dysku
%~p1 - rozwija %1 tylko do ścieżki
%~n1 - rozwija %1 tylko do nazwy pliku
%~x1 - rozwija %1 tylko do rozszerzenia pliku
%~s1 - rozwinięta ścieżka zawiera tylko krótkie nazwy
%~a1 - rozwija %1 do atrybutów pliku
%~t1 - rozwija %1 do daty/czasu pliku
%~z1 - rozwija %1 do rozmiaru pliku
%~$PATH:1 - przeszukuje katalogi wymienione w zmiennej środowiskowej PATH i rozwija %1 do pełnej nazwy dla pierwszej znalezionej. Jeśli nazwa zmiennej środowiskowej nie została zdefiniowana lub pliku nie znaleziono w wyszukiwaniu, modyfikator rozwija do pustego ciągu znaków. Modyfikatory mogą być łączone w celu uzyskania złożonych wyników, np.:
%~dp1 - rozwija %1 tylko do litery dysku i ścieżki
%~nx1 - rozwija %1 tylko do nazwy pliku i rozszerzenia

W powyższym przykładzie %1 można zastąpić innymi prawidłowymi wartościami. Składnia %~ jest zakończona prawidłową liczbą argumentów. Modyfikatory %~ nie mogą być używane z %*

5. Polecenia zbiorów wsadowych

[CALL] nazwa_pliku [parametry_wsadowe]
Polecenie to powoduje wykonanie programu wsadowego zawartego w pliku nazwa_pliku (z ewentualnymi parametrami). Po jego wykonaniu nastąpi powrót do następnego polecenia w bieżącym zbiorze wsadowym (uwaga: brak polecenia CALL spowodowałby przerwanie wykonywania bieżącego zbioru wsadowego, ale brak powrotu do niego po zakończeniu wykonywania wywoływanego zbioru).

Przy włączonych rozszerzeniach poleceń polecenie CALL akceptuje obecnie etykiety jako obiekt docelowy. Składnia jest następująca:

CALL :etykieta argumenty

Tworzony jest nowy kontekst pliku wsadowego z podanymi argumentami, a sterowanie jest przekazywane do instrukcji po określonej etykiecie. Musisz wydać polecenie "exit" dwukrotnie osiągając koniec pliku skryptu dwukrotnie. Po pierwszym odczycie końca pliku sterowanie powróci bezpośrednio za instrukcję CALL. Po drugim odczycie nastąpi zakończenie skryptu.

ECHO [ON | OFF | tekst]

ECHO ON powoduje włączenie wyświetlania tekstów poleceń zbioru wsadowego w czasie jego wykonywania (tryb domyślny),
ECHO OFF powoduje wyłączenie wyświetlania tekstów poleceń,
ECHO (bez parametru) powoduje wyświetlenie komunikatu „echo is off" lub „echo is on" w zależności od stanu faktycznego,
ECHO *tekst* powoduje wyświetlenie napisu *tekst* na ekranie,
ECHO.(kropka bez spacji) powoduje wyświetlenie pustego wiersza.
Dodatkowo każde polecenie w zbiorze wsadowym, które zostanie poprzedzone znakiem @ nie będzie wyświetlone bez względu na wcześniejsze użycie polecenia ECHO ON/OFF.

FOR %zmienna IN (zbiór) DO polecenie [parametry_polecenia]

Wykonuje określone polecenie dla wszystkich plików ze zbioru plików, gdzie:

%zmienna - określa parametr wymienny.
(zbiór) - określa zbiór - jeden lub kilka plików. Używanie symboli wieloznacznych jest dozwolone.
polecenie - określa polecenie, które ma być wykonane dla każdego pliku.
parametry_polecenia -określa parametry lub opcje dla określonego polecenia.

Aby użyć polecenia FOR w programie wsadowym, wpisz %%zmienna zamiast %zmienna. W nazwach zmiennych rozróżnia się małe i wielkie litery, tak więc %i różni się od %I. Przy włączonych rozszerzeniach poleceń obsługiwane są następujące dodatkowe formy polecenia FOR:

FOR /D %zmienna IN (zbiór) DO polecenie [parametry-polecenia]
Jeśli "zbiór" zawiera symbole wieloznaczne, to mają być poszukiwane odpowiednie nazwy katalogów, a nie nazwy plików.

FOR /R [[dysk:]ścieżka] %zmienna IN (zbiór) DO polecenie [parametry-polecenia]
Wędruje po drzewie katalogów poczynając od katalogu [dysk:]ścieżka, wykonując polecenie FOR w każdym katalogu tego drzewa. Jeśli po /R nie podano żadnego katalogu, to używany jest katalog bieżący.
Jeśli "zbiór" jest określony tylko przez pojedynczy znak kropki (.), to polecenie wyliczy tylko zawartość drzewa katalogów.

FOR /L %zmienna IN (początek,krok,koniec) DO polecenie [parametry-polecenia]
W tym przypadku zbiór jest ciągiem liczb od wartości "początek" do wartości "koniec", zmieniających się o wartość "krok". Tak więc (1,1,5) generuje ciąg 1 2 3 4 5, a (5,-1,1) generuje ciąg (5 4 3 2 1)

FOR /F ["opcje"] %zmienna IN (zbiórplików) DO polecenie [parametry-polecenia]
FOR /F ["opcje"] %zmienna IN ("ciąg") DO polecenie [parametry-polecenia]
FOR /F ["opcje"] %zmienna IN ('polecenie') DO polecenie [parametry-polecenia]

lub, jeśli obecna jest opcja usebackq:
FOR /F ["opcje"] %zmienna IN (zbiórplików) DO polecenie [parametry-polecenia]
FOR /F ["opcje"] %zmienna IN ('ciąg') DO polecenie [parametry-polecenia]
FOR /F ["opcje"] %zmienna IN (`polecenie`) DO polecenie [parametry-polecenia]

zbiórplików jest jedną lub wieloma nazwami plików. Każdy plik jest otwierany, odczytywany i przetwarzany przed przejściem do następnego pliku z tego zbioru. Przetwarzanie polega na odczytaniu pliku, podzieleniu go na poszczególne wiersze tekstu i podzieleniu każdego wiersza na zero lub więcej leksemów. Wówczas wywoływana jest główne polecenie pętli, przy czym jako wartości zmiennych są przypisywane ciągi znalezionych leksemów. Domyślnie opcja /F przekazuje pierwszy oddzielony spacją leksem z pierwszego wiersza każdego pliku. Puste wiersze są pomijane. Można zmienić domyślny sposób podziału wierszy określając opcjonalny parametr "opcje". Jest to ciąg umieszczony w cudzysłowie, który zawiera jedno lub więcej słów kluczowych określających różne parametry podziału. Dostępne są następujące słowa kluczowe:

eol=c - określa znak komentarza końca wiersza (tylko jeden)

skip=n - określa liczbę wierszy do pominięcia na początku pliku.

delims=xxx - określa zestaw ograniczników. Zastępuje domyślny zestaw ograniczników (spację i tabulator).

tokens=x,y,m-n - określa tokeny, które mają być przekazywane z każdego wiersza do głównego polecenia w każdej iteracji. Spowoduje to przydzielenie dodatkowych nazw zmiennych. m-n oznacza zakres, czyli tokeny od m-tego do n-tego. Jeśli ostatni znak ciągu tokens= jest gwiazdką, wówczas przydzielana jest dodatkowa zmienna, która otrzymuje pozostały tekst z wiersza po przydzieleniu ostatniego tokenu.

usebackq - określa, czy wymuszane są nowe semantyki, gdzie ciąg w odwrotnych apostrofach jest wykonywany jako polecenie, a ciąg w apostrofach jest literałem polecenia i zezwala na użycie cudzysłowów dla nazw plików w *zbiórplików*.

Przykłady, które mogą pomóc:

FOR /F "eol=; tokens=2,3* delims=, " %i in (mójplik.txt) do @echo %i %j %k

Analizuje każdy wiersz w pliku *mójplik.txt*, ignorując wiersze rozpoczynające się od średnika, przekazuje drugi i trzeci token z każdego wiersza do głównego polecenia, z tokenami rozdzielonymi przecinkami i/lub spacjami. Zauważ, że dla głównego polecenia odwołanie instrukcji %i pobiera drugi token, %j trzeci token, a %k pobiera wszystkie pozostałe tokeny po trzecim. Dla nazw plików zawierających spacje, należy stosować cudzysłowy. Aby używać cudzysłowów w ten sposób, należy również użyć opcji usebackq, bo w przeciwnym razie cudzysłowy będą interpretowane do analizy jako ciąg literalny. %i jest deklarowane jawnie w instrukcji, a %j i %k są jawnie deklarowane poprzez opcję tokens=. Można określić do 26 tokenów za pomocą wiersza tokens=, nie próbując zadeklarowania zmiennej większej niż litera 'z' lub 'Z'.
Pamiętaj, że nazwy zmiennej FOR uwzględniają wielkość liter, są globalne i nie można mieć ich aktywnych więcej niż 52 w tym samym czasie.

Można również użyć logicznego analizowania FOR /F na ciągu natychmiastowym umieszczając w apostrofach *zbiórplików* między nawiasami. Będzie to potraktowane jako pojedynczy wiersz wyjścia z pliku i przeanalizowane.

Na koniec, można też użyć polecenia FOR /F do analizy wyjścia polecenia. Dokonuje się tego umieszczając w odwrotnych apostrofach *zbiórplików* między nawiasami. Będzie to potraktowane jako wiersz polecenia, który jest przekazywany do podrzędnego programu CMD.EXE. Wyjście jest wtedy przechwytywane do pamięci i analizowane jakby był to plik. Zobacz następujący przykład:
FOR /F "usebackq delims==" %i IN (`SET`) DO @echo %i
wyliczy nazwy zmiennych środowiskowych w bieżącym środowisku. Ponadto, zostało ulepszone podstawianie odwołań zmiennej FOR, można teraz używać następującej opcjonalnej składni:

%~I - rozwija %I usuwając wszystkie obejmujące cudzysłowy (")
%~fI - rozwija %I do pełnej nazwy ścieżki
%~dI - rozwija %I tylko do litery dysku
%~pI - rozwija %I tylko do ścieżki
%~nI - rozwija %I tylko do nazwy pliku
%~xI - rozwija %I tylko do rozszerzenia pliku
%~sI - rozwinięta ścieżka zawiera tylko krótkie nazwy
%~aI - rozwija %I do atrybutów pliku
%~tI - rozwija %I do daty/czasu pliku
%~zI - rozwija %I do rozmiaru pliku
%~$PATH:I - przeszukuje katalogi wymienione w zmiennej środowiskowej PATH i rozwija %I do pełnej nazwy dla pierwszej znalezionej. Jeśli nazwa zmiennej środowiskowej nie została zdefiniowana lub pliku nie znaleziono w wyszukiwaniu, modyfikator rozwija do pustego ciągu.

Modyfikatory mogą być łączone w celu uzyskania złożonych wyników:

%~dpI - rozwija %I tylko do litery dysku i ścieżki
%~nxI - rozwija %I tylko do nazwy pliku i rozszerzenia
%~fsI - rozwija %I tylko do pełnej nazwy ścieżki z krótkimi nazwami
%~dp$PATH:i - przeszukuje katalogi wymienione w zmiennej środowiskowej PATH i rozwija %I do litery dysku i ścieżki dla pierwszej znalezionej.
%~ftzaI - rozwija %I do DIR jak wiersz wyjściowy

W powyższych przykładach %I i PATH można zastąpić innymi prawidłowymi wartościami. Składnia %~ jest zakończona prawidłową nazwą zmiennej FOR. Stosowanie dużych liter dla nazw zmiennych, jak np. %I czyni je czytelniejszymi i zapobiega myleniu z modyfikatorami, które nie uwzględniają wielkości liter.

GOTO etykieta
Powoduje przekazanie sterowania (skok) do linii zawierającej podaną etykietę. Za etykietę uważa się dowolną linię poprzedzoną znakiem **:** (dwukropka). Treść linii po etykiecie jest ignorowana. Przy włączonych poleceniach rozszerzeń polecenie GOTO akceptuje etykietę docelową `:EOF`, która przekazuje sterowanie na koniec bieżącego pliku skryptu wsadowego. Jest to łatwy sposób kończenia pliku skryptu wsadowego bez definiowania etykiety.

IF [NOT] warunek polecenie
Wykonuje przetwarzanie warunkowe w programach wsadowych. Powodujące wykonanie *polecenia* o ile jest spełniony podany *warunek*. Może on (*warunek*) być zapisany w następujący sposób:

ERRORLEVEL liczba - określa prawdę (spełnienie warunku), jeśli ostatnio wykonany program zwrócił kod błędu równy lub większy od podanej liczby,
ciąg1= =ciąg2 - określa prawdę (spełnienie warunku), jeśli podane ciągi tekstowe są identyczne,
EXIST nazwa_pliku - określa spełnienie warunku, jeśli plik o nazwie "nazwa_pliku" istnieje,

Po poleceniu może wystąpić słowo kluczowe ELSE, co spowoduje wykonanie polecenia znajdującego się po słowie kluczowym ELSE, jeżeli podany warunek nie zostanie spełniony. Konstrukcja ELSE musi występować w tym samym wierszu, co polecenie występujące po słowie IF. Na przykład:

IF EXIST nazwa_pliku (

```
    del nazwa_pliku
  ) ELSE (
    echo Brak pliku "nazwa_pliku"
  )
```

Jeżeli włączone są rozszerzenia poleceń, instrukcja IF zmienia się następująco:

```
IF [/I] ciąg1 operator_porównania ciąg2  polecenie
IF CMDEXTVERSION liczba  polecenie
IF DEFINED zmienna  polecenie
```

gdzie *operator_porównania* może być jednym z:

```
EQU - równe
NEQ - nie równe
LSS - mniejsze niż
LEQ - mniejsze niż lub równe
GTR - większe niż
GEQ - większe niż lub równe
```

a użycie przełącznika /I powoduje wykonanie porównania ciągów bez uwzględniania wielkości liter. Przełącznik /I może być również użyty przy porównaniu ciąg1==ciąg2. Są to porównania rodzajowe w tym sensie, że, jeżeli zarówno ciąg1, jak i ciąg2 składają się wyłącznie z cyfr, ciągi są konwertowane na liczby i wykonywane jest porównanie numeryczne. Wyrażenie warunkowe CMDEXTVERSION działa podobnie do zmiennej ERRORLEVEL, z wyjątkiem tego, że jest to porównanie z wewnętrznym numerem wersji skojarzonym z rozszerzeniami poleceń. Pierwszym numerem wersji jest 1. Wartość ta będzie zwiększana o 1, jeżeli do rozszerzeń poleceń dodane zostaną znaczące ulepszenia. Wyrażenie warunkowe CMDEXTVERSION nigdy nie jest prawdziwe, jeżeli rozszerzenia poleceń są wyłączone. Wyrażenie warunkowe DEFINED działa podobnie do wyrażenia EXISTS, z wyjątkiem tego, że argumentem jest nazwa zmiennej; wyrażenie jest prawdziwe, jeżeli zmienna środowiskowa jest zdefiniowana. Wyrażenie %ERRORLEVEL% jest rozwijane w reprezentację tekstową bieżącej wartości zmiennej ERRORLEVEL przy założeniu, że nie ma jeszcze zmiennej środowiskowej o nazwie ERRORLEVEL, w którym to przypadku pobrana zostanie jej wartość.

Wyrażenie **%CMDCMDLINE%** jest rozwijane do oryginalnej postaci wiersza polecenia, przekazanego do programu CMD.EXE, przed jego przetworzeniem, przy założeniu, że nie ma jeszcze zmiennej środowiskowej o nazwie CMDCMDLINE, w którym to przypadku pobrana zostanie jej wartość.

Wyrażenie **%CMDEXTVERSION%** jest rozwijane w reprezentację tekstową bieżącej wartości zmiennej CMDEXTVERSION przy założeniu, że nie ma jeszcze zmiennej środowiskowej o nazwie CMDEXTVERSION, w którym to przypadku pobrana zostanie jej wartość.

PAUSE
Powoduje wyświetlenie komunikatu „Press any key to continue” (lub równoważnego dla wersji zlokalizowanej systemu) i następuje zawieszenie wykonania programu aż do momentu naciśnięcia klawisza.

REM [tekst]
Polecenie służące do umieszczania wewnątrz zbiorów wsadowych komentarzy.

SHIFT [/n]
Polecenie to jest wykorzystywane jeśli podczas wywołania zbioru wsadowego podaliśmy dodatkowe parametry. Powoduje ono wtedy zmianę ich numeracji, tak że parametrowi %0 przypisywana jest wartość parametru %1, parametrowi %1 wartość %2, itd. Możliwy jest w ten sposób dostęp do większej liczby parametrów niż 9. Przy włączonych poleceniach rozszerzeń polecenie SHIFT obsługuje przełącznik /n, którego użycie powoduje, że polecenie przesuwa od n-tego argumentu, gdzie n może być z przedziału między zero a osiem. Na przykład:
SHIFT /2
przesunie %3 do %2, %4 do %3 itd. i pozostawi %0 oraz %1.

6. Filtry

Filtrami nazywamy programy, które pobierają dane wejściowe ze standardowego wejścia i wysyłają je po ewentualnym przetworzeniu na standardowe wyjście. Dzięki temu możliwe jest łączenie takich programów w potoki (*polecenie* | *polecenie*)

FIND [/V][/C][/N][/I] „*tekst*" [*zbiór*]
/V wysłanie na wyjście wierszy NIE zawierających podanego *tekstu* wzorca,
/C wysłanie na wyjście liczby linii zawierających podany wzorzec,
/N każdy wysyłany wiersz będzie opatrzony swoim numerem w zbiorze,
/I ignorowanie różnicy między małymi i dużymi literami.
Pominięcie nazwy zbioru powoduje wczytanie danych ze standardowego wejścia.
Przykład:
DIR | FIND „Volume in drive" >> LABELS.TXT
Dopisanie etykiety dysku do zbioru LABELS.TXT.

MORE
Polecenie pozwala na wyświetlanie tekstu na ekranie po stronie. Więcej: MORE /?
Przykład:
MORE < sorted.txt
Wyświetli na ekranie po stronie plik sorted.txt.

SORT [/R][/+n]
Umożliwia on sortowanie wierszy zbioru wejściowego.
/R sortowanie w odwrotnej kolejności
/+*n* sortowanie rozpoczynając od *n*-tego znaku w wierszu
Małe i duże litery nie są rozróżniane. Więcej: SORT /?
Przykład:
CON | SORT > sorted.txt
Wiersze wpisane z klawiatury (zakończone Ctrl-Z) zostaną posortowane i wpisane do zbioru sorted.txt.

7. Uruchamianie aplikacji z okna DOS w Windows

START
Uruchamia aplikacje Windows lub programy dosowe, dla których wywołuje kolejną sesję DOS-a. Polecenia tego można także użyć do otwarcia dokumentu, jeśli w środowisku Windows jest zdefiniowana skojarzona z tym dokumentem aplikacja. Na przykład start info.txt uruchomi NOTEPAD-a otwierając w nim zbiór info.txt.
START [options] [program] [arg...]. Dokładny opis parametrów można uzyskać po wydaniu w wierszu poleceń: START /?.

Uruchamianie aplikacji windows z okna DOS-a
W oknie Dos-a można uruchamiać aplikacje windows podając nazwę uruchamianego programu i ewentualnie parametry jego wywołania. Aplikacje windows można także uruchamiać ze zbiorów BAT.

Technika „Drag & Drop" w oknie DOS-a
Do okienka dosowego można przeciągnąć ikony plików i folderów, powoduje to umieszczenie nazwy plików lub folderów, których ikony przeciągamy w buforze klawiatury sesji dosowej.

Kopiowanie i przesuwanie informacji do i z okna DOS-a
Jeśli jest aktywna belka narzędziowa w okienku DOS-a można zaznaczyć obszar ekranu i skopiować go lub wyciąć a następnie umieścić w aplikacji windows komendą WKLEJ. Można także użyć klawiszy Ctrl-C i Ctrl-V. Operacja przemieszczania informacji działa także w drugą stronę, to znaczy do okna DOS-owego z aplikacji windows lub innego okna DOS-owego.

Dodatkowe skróty w oznaczaniu katalogów
. oznacza bieżący katalog (znane ze starego DOS-a)
.. oznacza nadrzędny katalog do bieżącego (znane ze starego DOS-a)
... oznacza katalog o dwa poziomy wyżej od bieżącego (nowość od DOS-Win95)
.... oznacza katalog o trzy poziomy wyżej od bieżącego (nowość od DOS-Win95)

8. „Shortcut-y" dla aplikacji DOS-a

Dla aplikacji Dos-owych (również BAT-ów) można tworzyć „shortcut-y" na „desktop-ie" i w „start menu". Sposób uruchomienia takiego programu i jego zachowanie można modyfikować za pomocą opcji „properties". Modyfikacji mogą podlegać: atrybuty pliku, linia wywołania programu wraz z parametrami,katalog roboczy programu, rodzaj okna w którym program ma się uruchomić, czcionka która ma być używana przez program, wielkość i rodzaj przydzielanej programowi pamięci przez system operacyjny, ikona programu, wygląd i zachowanie okna z programem.

9. Podpowiedzi

Jak sprawdzić czy istniej katalog ? *Nul* jest urządzeniem/plikiem systemowym zawsze obecnym po poprawnym uruchomieniu systemu, dlatego możemy badać jego obecność w katalogu:

```
IF EXIST C:\TEMP\nul GOTO :jest
```

Najprostszy edytor ? Gdy wszystko zawiedzie można użyć polecenia:

```
COPY con nazwa_pliku
```

po wydaniu polecenia wszystkie znaki wprowadzone z konsoli (urządzenie/plik CON) zostaną skopiowane do pliku. Ctrl+Z kończy wprowadzanie i zamyka plik. Uwaga: w tym „edytorze" nie można się pomylić – mamy tylko możliwość usuwania znaku (BACKSPACE) w jednej linii.

Przejście do katalogu, którego nazwa zawiera spację, np: C:\Progam Files\. Nazwę należy nazwę ująć w cudzysłowy:

CD „C:\Program Files”

Usunięcie zmiennej środowiska można uzyskać prze wydanie polecenia:

SET nazwa_zmiennej=

Uruchamianie zbioru wsadowego. Gdy wystąpią problemy z programem należy w kluczowych punktach skryptu wykorzystać polecenie *pause* do zatrzymania jego wykonywania i polecenie *set* oraz *echo* do podglądania parametrów i zmiennych systemowych.

10. Literatura i dodatkowe informacje

[1] Szczegółowe informacje o poszczególnych poleceniach zewnętrznych i wewnętrznych dostępnych z linii komend można uzyskać wydając polecenie: *nazwa_polecenia* /? lub HELP *nazwa_polecenia.*
[2] Pomoc zawarta w systemie Windows 2000 (Start->Pomoc), temat: „Strona główna spisu poleceń systemu Windows 2000”.
[3] „Polecenia Windows 2000 – leksykon kieszonkowy”, Aeleen Frish, Helion 2001, ISBN:837197-561-9
[4] Oficjalna witryna Microsoft: http://www.microsoft.com/windows2000/
[5] Godne polecenia: Podręcznik on-line poleceń Windows NT/2000/XP: http://www.ss64.demon.co.uk/nt/index.html
[6] Zaawansowane narzędzia i programy pomocnicze do Windows 2000 (i nie tylko) http://www.sysinternals.com

Przygotowanie do laboratorium:

1. Jakie znasz rodzaje pośrednictwa użytkownika w systemach operacyjnych ? Jakie są ich cechy ?
2. Co jest (jaki program) domyślnym pośrednikiem trybu tekstowego w Windows 2000 ? Czy znasz inne programy, które mogą pełnić tę rolę ? Co to są polecenia wewnętrzne i zewnętrzne ?
4. Zapoznaj się z różnymi możliwościami użycia polecenia START. Sprawdź co można zrobić używając techniki „Drag & Drop” w oknie DOS.
5. Zapoznaj się z poleceniem SETLOCAL i ENDLOCAL. Jaka jest widoczność zmiennych środowiskowych ?
6. Zapoznać się dokładnie z rozszerzoną składnią polecenia FOR. Szczególnie z opcjami FOR /I, FOR /L i FOR /F.
7. Zapoznać się z mechanizmem opóźnionego rozwijania zmiennych (opis w poleceniu IF).
8. Dla wprawy napisać program wsadowy, który wypisze wszystkie podane parametry w postaci ponumerowanej listy. Lista parametrów może być dowolna, także pusta.

Np. wywołanie:

mój.bat A 123 BC DE zyzio

wypisze:

par1=A
par2=123
par3=BC
par4=DE
par5=zyzio