ROZDZIAŁ 4
Strumienie danych, potoki danych oraz pliki wsadowe.

Strumienie danych

W systemie Windows ekran monitora jest traktowany jako standardowe urządzenie wyjściowe. Klawiatura pełni rolę standardowego urządzenia wejściowego. Określane są jako konsola (CON). Niestandardowymi strumieniami są również:

- drukarka(PRN),
- złącza szeregowe (COM1 do COM4),
- złącza równoległe (LPT1 do LPT3),
- urządzenie puste (NUL),
- synonim nazwy COM1 (AUX).

Urządzeniem logicznym NULL jest urządzenie, któremu nie odpowiada żadne urządzenie fizyczne. Zapisywanie do urządzenia pustego nie powoduje żadnych efektów, natomiast odczytywanie powoduje sygnalizację końca zbioru.

Przekierowania w Windows 7

W systemie Windows można skierować wyjście programu na inny strumień oraz pobrać wejście z innego.
> (znak większości) przekierowuje dane z konsoli do pliku
a.bat > plik1.txt - przekierowuje wyjście pliku wsadowego a.bat do pliku plik1.txt.

>> (dwa znaki większości) wyjście zostaje dopisane do pliku, jeśli taki już istnieje
a.bat >> plik.txt - utworzy plik plik.txt i skieruje do niego wyjście pliku a.bat. Jeśli plik.txt istnieje, wyjście pliku a.bat zostanie do niego dopisane

2> przekierowuje tylko wyjście błędów
a.bat 2> error.fil - wysyła wszelkie błędy występujące podczas uruchomienia pliku a.bat do pliku error.fil

2>> tak jak 2>, lecz wyjście zostanie dodane do istniejącego pliku

< (znak mniejszości) podaje dane ze źródła na wejście polecenia
[polecenie] < plik.txt
otwierany plik.txt jako dane wejściowe do programu [polecenie]. Aby umieścić w nowym pliku posortowany plik.txt wpisz:

sort <plik.txt> posortowany.txt

Przetwarzanie potokowe

Przetwarzaniem potokowym nazywamy skierowanie strumienia danych wyjściowych jednego polecenia do innego polecenia jako jego strumienia danych wejściowych.

| - (tzw. potok) przesłanie standardowego wyjścia **polecenia1** jako standardowego wejścia do **polecenia2.**

polecenie1 | polecenie2

Polecenia (filtry) stosowane w przetwarzaniu potokowym :

• Polecenie **more** wyświetla kolejno pojedyncze strony zawartości pliku lub danych wyjściowych innego polecenia.
• Polecenie **find** wyszukuje w plikach lub danych wyjściowych innych poleceń znaki określone przez użytkownika.
• Polecenie **sort** sortuje zawartość plików lub danych wyjściowych innych poleceń.

More

Polecenie **More** wyświetla zawartości pliku ekran po ekranie.

MORE [/E [/C] [/P] [/S] [/Tn] [+n]] < [dysk:][ścieżka]plik
nazwa_polecenia | MORE [/E [/C] [/P] [/S] [/Tn] [+n]]
MORE /E [/C] [/P] [/S] [/Tn] [+n] [pliki]

[dysk:][ścieżka]plik - Określa plik, który ma być wyświetlany po jednym ekranie na raz.
nazwa_polecenia - Określa polecenie, którego wynik ma być wyświetlany po jednym ekranie na raz.

/E – Włączane są rozszerzone funkcje.
/C – Przed wyświetleniem strony czyszczony jest ekran.
/P – Znaki nowego wiersza są rozszerzane.
/S - Łączy sąsiednie puste wiersze w jeden wiersz.
/Tn – Zmiana tabulatorów na spacje (domyślnie 8).
+n - Zaczyna wyświetlanie pierwszego pliku od wiersza n.
Pliki – Zawiera listę plików do wyświetlenia. Pliki z listy są oddzielane pustymi wierszami.

Jeśli włączone są funkcje rozszerzone, to akceptowane będą następujące polecenia:

P n - Wyświetla następne n wierszy
S n - Pomija następne n wierszy
F - Wyświetla następny plik
Q - Koniec
= - Wyświetla numer wiersza
? - Wyświetla wiersz pomocy
<spacja> - Wyświetla następną stronę
<enter> - Wyświetla następny wiersz

Find

Polecenie **Find** umożliwia wyszukanie określonych ciągów znaków w pliku.

*Uwaga. **Find** domyślnie rozróżnia małe i duże litery.*

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]]" ciąg" [[dysk:][ścieżka]plik[...]]

/V - Pokazuje wszystkie wiersze NIE zawierające podanego ciągu.
/C - Pokazuje tylko liczbę wierszy zawierających ciąg.
/N - Pokazuje wiersze i ich numery.
/I - Nie rozróżniane są małe i duże litery podczas wyszukiwania ciągów.
/OFF[LINE] - Nie pomija plików z ustawionym atrybutem przesunięcia.
„ ciąg" – Określany jest ciąg tekstowy do znalezienia.
[dysk:][ścieżka]plik – Określany jest pliki do przeszukiwania.

*Uwaga: Jeżeli pominięto nazwę pliku, polecenie **find** działa jako filtr, pobiera dane wejściowe ze standardowego źródła danych wejściowych (zazwyczaj klawiatury, potoku lub przekierowanego pliku), a następnie powoduje wyświetlenie wszystkich wierszy zawierających ciąg określony przez parametr ciąg.*

Przykład:

Utwórz plik o nazwie dni1.txt o zawartości:

aPoniedziałek
aWtorek
aŚroda
aCzwartek
aPiątek
aSobota
aCzwartek
aNiedziela
aCzwartek
aPiątek
aPoniedziałek

Wyświetl numery wierszy w których występuje aCzwartek.

Aby wykonać zadanie:

1. Uruchom okno konsoli
2. Wpisz w wierszu poleceń : **find /n ''aCzwarte''**
3. Zapoznaj się z informacjami na ekranie

Sort

Polecenie **Sort** sortuje wiersze zapisane w pliku.

SORT [/R] [/+n] [/M KB] [/L ustawienia regionalne] [/REC bajty rekordu]
[[dysk1:][ścieżka1]plik1] [/T [dysk2:][ścieżka2]] [/O [dysk3:][ścieżka3]plik3]

/+n – wskazuje numer znaku, od którego ma się rozpoczynać każde porównywanie.
/+3 określa, że każde porównywanie powinno się rozpoczynać od trzeciego znaku w każdym wierszu. Wiersze krótsze niż n znaków są sortowane przed innymi wierszami. Domyślnie, porównania rozpoczynają się od pierwszego znaku każdego wiersza.
/L[OCALE] - ustawienia regionalne Zastępuje domyślne ustawienia regionalne systemu określonymi ustawieniami. Ustawienie" „ C" „ daje najszybsze sortowanie i jest obecnie jedyną możliwością. W sortowaniu nigdy nie jest uwzględniana wielkość liter.
/M[EMORY] KB - wskazuje ilość pamięci głównej do użycia w sortowaniu, w kilobajtach. Wielkość pamięci jest zawsze ograniczona od dołu wartością 160 KB. Jeżeli podana zostanie wielkość pamięci, w sortowaniu użyta zostanie dokładna ilość pamięci, bez względu na ilość dostępnej pamięci głównej.
/REC[ORD_MAXIMUM] znaki - wskazuje maksymalną liczbę znaków w rekordzie (domyślnie 4096, maksymalnie 65535).
/R[EVERSE] - Odwraca porządek sortowania, to znaczy sortuje od Z do A, następnie od 9 do 0.
[dysk1:][ścieżka1]plik1 - wskazuje plik do posortowania. Jeżeli plik nie zostanie określony, użyte zostanie standardowe wejście. Określenie pliku wejściowego jest szybsze niż przekierowywanie tego samego pliku na standardowe wejście.
/T[EMPORARY][dysk2:][ścieżka2] - wskazuje ścieżkę katalogu, w którym mają być zapisywane pliki robocze sortowania, w przypadku gdy dane nie zmieszczą się w pamięci głównej. Domyślnie używany jest systemowy katalog tymczasowy.
/O[UTPUT] [dysk3:][ścieżka3]plik3 - wskazuje plik, w którym mają być zapisane posortowane dane wejściowe. Jeżeli plik nie zostanie określony, dane będą zapisywane do standardowego wyjścia. Wskazanie pliku wyjściowego jest szybsze niż przekierowanie standardowego wyjścia do tego samego pliku.

*Uwaga: W poleceniu **sort** wielkie i małe litery są rozróżniane.*

Przykład:

Utwórz plik o nazwie dni.txt o zawartości:

1. aPoniedziałek
2. aWtorek
3. aŚroda
4. aCzwartek
5. aPiątek
6. aSobota
7. aNiedziela

Posortuj w kolejności alfabetycznej dni. Wynik zapisz do pliku dni_sort.txt

Aby wykonać zadanie:

1. Uruchom okno konsoli
2. Wpisz w wierszu poleceń : **sort /+4 dni.txt /o dni_sort.txt**
3. Porównaj wynik sortowani za pomocą polecenia: type dni_sort.txt

Wbudowane zmienne środowiskowe

W systemie Windows występują tzw. zmienne środowiskowe. W czasie ich użycia są rozwijane do ich zawartości. Przykładowe wbudowane zmienne środowiskowe:

%username% - Nazwa bieżącego użytkownika (małymi literami).
%homedrive% - Jest rozwijana w literę dysku na którym znajduje się katalog macierzysty bieżącego użytkownika.
%homepath% - Jest rozwijana w ścieżkę katalogu macierzystego bieżącego użytkownika.
%homeshare% - Jest rozwijana w zasób dzielony zawierający katalog macierzysty bieżącego użytkownika.
%processor architecture% - Jest rozwijana w słowo kluczowe zawierające producenta procesora zainstalowanego w systemie (np. x86, AMD64 lub alpha).
%processor level% - Jest rozwijana w liczbę wskazującą model procesora z danej rodziny.
%errorlevel% - obejmuje stan zakończenia ostatniego polecenia. Wartość 0 generalnie wskazuje poprawne wykonanie polecenia, a wartość 1 wskazuje błąd. Niektóre polecenia wykorzystują zmienną do określenia wartości, której interpretacja jest zależna od polecenia.
%windir% - Zwraca lokalizację katalogu systemu operacyjnego.
%os% - Zwraca nazwę systemu operacyjnego.
%prompt%- Zwraca ustawienia wiersza poleceń.
%number_of_processors% - Określa liczbę procesorów (rdzeni) zainstalowanych w komputerze.

Aby wyświetlić pełną listę zmiennych środowiskowych wpisz polecenie **set.**

Pliki wsadowe (skrypty)

Skrypt jest to plik tekstowy do wielokrotnego wykorzystania w którym zawarte są polecenia. Skrypty uruchamia się w taki sam sposób jak programy wykonywalne. Po uruchomieniu skryptu, wykonywane są zawarte w nim polecenia, w takiej samej kolejności w jakiej zostały zapisane jedno po drugim.

Argumenty skryptu

Każdy pliki wsadowy morze pobierać argumenty. Wewnątrz programu wsadowego argumenty jego wywołania reprezentowane są przez napis %n, gdzie n oznacza numer argumentu. Wywołajmy następujące polecenie:
Plik.bat a b c d
Napis plik.bat jest to nazwa pliku wsadowego, a b c d są przekazywanymi mu argumentami. W pliku wsadowym napis **a** będzie reprezentowany **jako %1, b jako %2, c jako %3 i d jako %4.** Gdyby plik ten zawierał instrukcję
Echo %1

Zostałaby ona przetworzona jak instrukcja echo a, a jej wynikiem byłoby wypisanie na ekranie słowa a. Analogicznie jeżeli plik.bat zawierałby instrukcję
Echo %4 %3 %2 %1
zostałaby ona zinterpretowana jako echo d c b a wynik taki pojawiłby się na ekranie.

Echo – na ekranie wyświetlany jest tekst podany jako argument. Można również użyć polecenia echo on do wyświetlania na ekranie poleceń skryptu, albo echo off do zakazu wyświetlania poleceń, wówczas wyświetlane są tylko wyniki działania poleceń. Pokazuje to poniższy przykład:

C:\>copy con test
Echo on
Rem pierwszy test echo on
Echo wynik pierwszego testu
Echo off
Rem drugi test ale z echo off
Echo wynik testu drugiego

C:\>test.bat { uruchamianie skryptu }
C:\>echo on
C:\>rem pierwszy test echo on
C:\>echo wynik pierwszego testu
Wynik pierwszego testu
C:\>echo off
Wynik testu drugiego

C:\>_

Zmienne

W plikach o rozrzedzeniu BAT można definiować własne zmienne, które mogą być użyte do dowolnych celów określonych przez programistę. W skrypcie zmienne mogą być definiowane za pomocą polecenia set, aby zapoznać się z szczegółową składnią polecenie set wpisz: **help set.** Użycie bez parametru polecenia set spowoduje wyświetlenie bieżących ustawień zmiennych środowiskowych.
Np.

Przypisujemy nowej zmiennej ścieżkę c:\dane i wyświetlamy jej wartość
C:\>set nowy =c:\dane & echo %nowy%
c:\dane
Odwołane do zmiennych następuje przez otoczenie ich nazwy znakami %.
Opcja /A polecenia set może być zastosowana do przypisania zmiennej wyniku wyrażenia arytmetycznego np.
C:\Set z = 10
C:\Set /A z = %z% + 1 - zwiększenie z o 1

Polecenia Setlocal i Endlocal są używane do ograniczenia zasięgu lokalnych zmiennych w skryptach.

Działanie Shift

Za pomocą polecenie shift zmienia się kolejność parametrów przekazywanych do programu wsadowego. Shift bez żadnych opcji przesuwa każdy argument o jedną pozycję w lewo - do początku listy argumentów - powodując, że drugiemu argumentowi wiersza poleceń odpowiada %1, trzeciemu %2 i tak dalej. Jeżeli shift użyto z opcją liczbową, przesuwane są jedynie argumenty o numerach równych lub większych od tej liczby. W ten sposób shift /4 rozpoczyna przesuwanie argumentów od %4, pozostawiając %1, %2 i %3 bez zmian. Oto prosty skrypt ilustrujący te zasady:

```
C:\>type przesuń.bat
echo off
echo nazwa: %0
echo linia l: %1 %2 %3 %4 %5 %6 %7
shift
echo linia 2: %1 %2 %3 %4 %5 %6 %7
shift /4
echo linia 3: %1 %2 %3 %4 %5 %6 %7
echo wszystkie argumenty: %*

C:\>przesuń 1 2 3 4 5 6 7 8 9

C:\>echo off
linia l: 1 2 3 4 5 6 7
linia 2: 2 3 4 5 6 7 8
linia 3: 2 3 4 6 7 8 9
wszystkie argumenty: 1 2 3 4 5 6 7 8 9
```

Zmienna **%*** oznacza oryginalny zestaw argumentów wiersza poleceń (niezmieniony przez operację shift). **%0** zawiera nazwę polecenia, pierwszy element w wierszu polecenia. Argumenty mogą być modyfikowane przed użyciem przez umieszczenie sposobu modyfikacji pomiędzy znakiem **%** a numerem argumentu. Na przykład postać **%~d3** oznacza literę dysku z trzeciego argumentu. Modyfikacje są zawsze określane tyldą, za którą następuje kod modyfikacji, składający się z jednej lub kilku liter.

Dostępne są następujące modyfikatory:

%~fn Pełna ścieżka.
%~dn Tylko litera dysku.
%~pn Tylko katalog.
%~nn Tylko nazwa pliku.
%~xn Tylko rozszerzenie pliku.
%~sn Używanie nazw 8.3, a nie długich nazw ścieżek
% ~$PATH:n Sprawdzanie ścieżki poszukiwań w zmiennej środowiskowej PATH i zwracanie pełnej ścieżki dla pasującego polecenia podanego jako argument n oraz zwracanie pustego ciągu znaków, jeżeli element nie zostanie znaleziony.

Call

Za pomocą polecenia **call** wywołuje się inny plik wsadowy, a następnie po jego zakończeniu zwraca sterowania do pliku wywołującego
Call ul.cmd
Wywoływany jest plik ul.cmd z wnętrza innego pliku wsadowego, do którego następnie zwraca sterowanie.

Goto

Instrukcja skoku bezwarunkowego **GOTO** to służy do ustalenia kolejności wykonywania instrukcji w plikach wsadowych
GOTO etykieta

Etykieta definiuje wiersz w programie wsadowym, do którego powinno zostać przekazane sterowanie. Deklarując etykietę poprzedza się ją dwukropkiem, np :etykieta.
Przykład:

```
C:\>type rodzina.bat
echo off
echo P1
goto mama
echo P2
:mama
echo P3

call ul
echo P4
C:\>type dom.bat
echo ul1
goto: EOF
echo ul2
C:\>rodzina


C:\>echo off
P1
P3
ul1
P4
```

Pętle

Język skryptów posiada kilka konstrukcji pętli FOR oto dwie, które są najbardziej popularne:

- Tradycyjna pętla indeksowana

For /L %%zmienna IN (start, skok, koniec) do polecenie

Podany parametr jest inicjowany przez wartość start, zwiększany o wartość skok za każdym wykonaniem pętli, pętla kończy działanie, gdy przekroczy limit wyznaczony przez koniec. Opcja **/L** – iteruje ciąg numeryczny

- Pętla dla wszystkich elementów listy

for %%zmienna in (zestaw) do polecenie

każdemu elementowi z listy kolejno jest przypisywany parametr.

Parametr **%%zmienna** może być wykorzystany do wykonania polecenia **for** w programie wsadowym. Parametr **%zmienna** może być wykorzystany do wykonania polecenia **for** z wiersza polecenia.

Przykład:

```
E:\>type pętla.bat
ECHO OFF
for /L %%1 in (1,1,5) do echo %%1
rem komentarz
E:\>pętla
E:\>ECHO OFF
1
2
3
4
5
```

Instrukcje warunkowe

Instrukcja warunkowa występuje w kilku wariantach.

IF [not] errorlevel poziom polecenie

Testowany jest kod wyjściowy poprzedniego polecenia. Jeżeli polecenie zakończyło się sukcesem zostanie zwrócony kod o wartości zero, jeśli zostanie zwrócona inna wartość niż zero oznacza to że wystąpił błąd podczas wykonywania polecenia.

Warunek jest prawdziwy, jeśli poprzednie polecenie zwróciło kod błędu równy bądź większy od podanego poziomu. Jeśli chcemy sprawdzić różne kody błędów musimy sprawdzać je we właściwej kolejności. Kod błędu ostatnio wykonywanego polecenia jest również przechowywany w zmiennej %ERRORLEVEL%

IF [not][/I] łańcuch1==łańcuch2 polecenie

Instrukcja ta sprawdza czy dwa łańcuchy są sobie równe. Jeśli zawierają spacje lub są nazwami zmiennych muszą być otoczone cudzysłowami. Jeśli porównujemy wartości parametrów, które mogą już zawierać cudzysłowy, należy otoczyć łańcuchy innymi znakami ograniczającymi np. nawiasami klamrowymi Opcja **/I** oznacza, że wielkość liter w porównywanych łańcuchach nie jest brana pod uwagę.

IF [not] exist plik polecenie

Instrukcja ta sprawdza czy istnieje określony plik. Można wykorzystać pełne ścieżki i wzorce nazw. Jeśli zostanie określony wzorzec to warunek jest spełniony, gdy istnieje przynajmniej jeden plik pasujący do wzorca. Można także sprawdzać istnienie katalogu podając jego nazwę w argumencie plik.

IF [/I] wartość1 op wartość2 polecenie

Instrukcja porównuje liczby albo łańcuchy. Dwie wartości są porównywane za pomocą operatora. Dozwolone są następujące operatory porównania:
EQU == (dwa znaki równości); dwa ciągi znaków są równe.
NEQ Dwa ciągi znaków są różne.
LSS Pierwszy ciąg znaków jest leksykalnie mniejszy niż drugi.
LEQ Pierwszy ciąg znaków jest leksykalnie mniejszy lub równy drugiemu.
GTR Pierwszy ciąg znaków jest leksykalnie większy od drugiego.
GEQ Pierwszy ciąg znaków jest leksykalnie większy lub równy drugiemu.

Zadania do samodzielnego wykonania

1. Przetestuj wszystkie przykładowe skrypty zawarte w ćwiczeniu.
2. Zapisz w pliku aczwartek.txt numery wierszy i wiersze w których występuje aCzwartek.
3. Napisz skrypt, który posortuje pliki alfabetycznie w katalogu bieżącym
4. Napisz skrypt, który zlokalizuje uszkodzone woluminy.
5. Zaproponuj własne pliki wsadowe, korzystając ze wszystkich znanych Ci poleceń i umieść je w sprawozdaniu.
6. Napisz skrypt, który zatrzymuje bufor wydruku czyści go i ponownie go uruchamia.
7. Napisz skrypt, który kopiuje pliki z jednego folderu do drugiego
8. Napisz skrypt, który będzie zakładał katalogi o nazwie : 1,2,3,4,5,6 oraz wyświetli czy zostały utworzone.

9. Napisz program, który będzie usuwał z dysku usb wszystkie pliki i foldery.
10. Napisz skrypt, który będzie wyświetlał informację o wersji systemu operacyjnego.
11. Napisz skrypt który będzie odzyskiwał dane z dysku USB. Użytkownik musi podać nazwę pliku do odzyskania.
12. Napisz skrypt, który zmienia system plików z Fat na NTFS
13. Napisz skrypt, który będzie wyświetlał mac karty sieciowej oraz konfigurację Ip.
14. Napisz skrypt, który wyświetla informacje o serwerze, stacji roboczej, grupach
15. Sprawdź działanie skryptu logowania na zajęciach poświęconym użytkownikom i grupom (ćwiczenie nr 6). Znając składnie poleceń napisz w punktach co robi nasz przykładowy skrypt logowania skrypt.bat
16. Napisz skrypt kolejnosc.bat, który wypisze liczby w kolejności 5,4,3,2,1 oraz 75,50,25,0 (zastosuj instrukcję pętli for).
17. Napisz skrypt o nazwie laboratorium.bat, który wyświetli 5 razy słowo laboratorium.
18. Napisz skrypt, który wyświetli nazwę bieżąco zalogowanego użytkownika, literę dysku i ścieżkę na której znajduje się katalog macierzysty bieżącego użytkownika, wyświetl producenta procesora zainstalowanego w systemie.
19. Napisz skrypt o nazwie stare.bat, który wyświetli listę wszystkich nazw plików z rozszerzeniem txt we wszystkich katalogach na dysku C, następnie znajdzie pliki mające nazwę Readme, posortuje je od drugiego znaku i wynik zapisze do pliku stare.txt. Następnie wywołaj skrypt laboratorium (z pkt.4). Wyświetl bieżącą datę bez wyświetlenia monitu o nową datę i również zapisz ją do pliku stare.txt. Na końcu wyświetl napis ”koniec poszukiwań“.
20. Napisz skrypt o nazwie przypisz.bat, który pod zmienną b przypisze słowo” blok“ a pod zmienną d słowo” krok“ jeśli zmienne nie mają różnych ciągów znaków to wyświetl napis” różne ciągi znaków. Natomiast jeśli pierwszy ciąg znaków (przypisany zmiennej b) jest leksykalnie mniejszy niż drugi (przypisany zmiennej d) to wyświetl napis ”pierwszy jest mniejszy“.
21. Utwórz plik plik.txt z dowolną zawartością. Następnie napisz procedurę wyszukaj.bat, która ma za zadanie znaleźć ciąg znaków z plik.txt określony jako pierwszy parametr wywołania procedury, wynik zapisz do pliku o nazwie będącej drugim parametrem wywołania procedury (np. wynik.txt).
22. Napisz procedurę sortowanie.bat, która posortuje plik podany jako pierwszy parametr wywołania procedury według drugiego znaku w każdym wierszu i wynik zapisze do pliku podanego jako drugi parametr wywołania procedury.
23. Zaproponuj własne pliki wsadowe, korzystając ze wszystkich znanych Ci poleceń i zamieść je w sprawozdaniu.

Uwaga ! Wszystkie napisane samodzielnie na zajęciach skrypty proszę zamieścić w sprawozdaniu.