## Programowanie wsadowe DOS/Windows – Część II

## 1. Szyfrowanie plików i katalogów

Do szyfrowania służy polecenie **cipher** o następującej składni:

cipher [{/e|/d}] [/s:katalog] [/a] [/i] [/f] [/q] [/h] [/k] [/u[/n]] [nazwa_ścieżki [...]] | [/r:nazwa_ścieżki_bez_rozszerzenia] | [/w:nazwa_ścieżki]

Najważniejsze z przełączników:

- /E (encrypt) - szyfrowanie katalogów
- /D (decrypt) - odszyfrowanie katalogów
- /A - uwzględnienie plików oraz katalogów
- /K - tworzenie nowego klucza szyfrowania
- /U - aktualizacja klucza szyfrowania
- /X - tworzenie kopii certyfikatu EFS i kluczy
- /Y - wyświetlenie znacznika certyfikatu EFS

Wywołanie polecenia **cipher** bez parametrów (lub z podaniem ścieżki) wyświetli status plików.

**Zadanie 1.**

Utworzyć dowolny plik tekstowy. Zaszyfrować go za pomocą polecenia **cipher**. Sprawdzić czy plik jest poprawnie zaszyfrowany (polecenie **cipher** bez przełączników). Spróbować otworzyć zaszyfrowany plik. Kto może otwierać takie pliki?

**Zadanie 2.**

Zastanowić się nad działaniem następującego skryptu:

```
@echo off
setlocal EnableDelayedExpansion
set encr=0
for /F %%i in ('cipher') do (
  if %%i EQU E (
    set /A encr=!encr!+1
  )
)
echo Zaszyfrowanych: %encr%
```

## 2. Kopiowanie plików i katalogów

Kopiowanie plików i katalogów możliwe jest za pomocą trzech komend: **copy** (kopiowanie i łączenie plików – z wyłączeniem katalogów), **xcopy** (kopiowanie plików i katalogów), **robocopy** (nowsza wersja xcopy).

Przykłady użycia:

- copy a.txt+b.txt c.txt - połączenie plików a.txt i b.txt, wynikiem jest plik c.txt
- xcopy C:\testA\ C:\testB\ /S - kopiowanie katalogu bez przenoszenia pustych podkatalogów
- robocopy C:\testA\ C:\testB\ *.txt - kopiowanie tylko plików z rozszerzeniem .txt

**Zadanie 3.**

Wyświetlić pomoc dotyczącą komend kopiowania. Sprawdzić czy istnieją przełączniki umożliwiające nadawanie atrybutów i kopiowanie plików zaszyfrowanych na nośniki nie umożliwiające szyfrowania.

## 3. Atrybuty plików i katalogów

Nadawanie i wyświetlanie atrybutów plików i katalogów umożliwia polecenie **attrib**:

attrib [+r|-r] [+a|-a ] [+s|-s] [+h|-h] [+i|-i] [drive:][path][filename] [/s [/d] [/l]]

- r - atrybut tylko do odczytu
- a - atrybut pliku archiwalnego
- s - atrybut pliku systemowego
- h - atrybut pliku ukrytego
- i - atrybut pliku nie indeksowalnego
- /s - przetwarzanie plików w podfolderach
- /d - uwzględnienie również folderów
- /l - uwzględnienie atrybutów łącz symbolicznych

Przykład wykorzystania polecenia **attrib**:

```
attrib +r +a -h plik.txt
```

Wywołanie polecenia **attrib** bez przełączników wyświetla informacje o plikach/katalogach, np.:

```
attrib C:\Windows\*
```

**Zadanie 4*** (rozwiązanie zadań oznaczonych * należy umieścić w sprawozdaniu lub pokazać podczas zajęć).

Napisać skrypt, który:

- Przyjmuje parametr – katalog
- Wyświetla wszystkie pliki tylko do odczytu z danego katalogu (wykorzystać polecenie **dir**)
- Pyta użytkownika czy: 0 – usunąć atrybut tylko do odczytu z wszystkich plików w tym folderze
  1 – stworzyć kopię tych plików w folderze C:\temp\ (polecenie **robocopy**)
  2 – zakończyć działanie skryptu

## 4. Wyszukiwanie tekstu w plikach

Wyszukiwanie tekstu umożliwia polecenie **find**:

find [/v] [/c] [/n] [/i] [/off[line]] "string" [[drive:][path]filename[...]]

- string - wyszukiwany ciąg znaków
- /v - wyświetlenie linii nie zawierających danego ciągu znaków
- /c - wyświetlenie tylko liczby linii z pasującym ciągiem znaków
- /n - wyświetlenie numeru linii i pasującego ciągu
- /i - ignorowanie wielkości znaku

Przykład:

```
find "Invalid" C:\Windows\* /N
```

**Zadanie 5.**

Uruchomić komendę z powyższego przykładu i zastanowić się nad jego działaniem. Uruchomić komendę również z przełącznikiem /i.

## 5. Potoki

Przetwarzaniem potokowym nazywamy skierowanie strumienia danych wyjściowych jednego polecenia do innego polecenia jako jego strumienia danych wejściowych. Składnia przedstawia się następująco:

polecenie1 | polecenie2

Wynik polecenia 1 zostanie przekazany do polecenia 2.

Przetwarzanie potokowe umożliwia łączenie **szerokiej gamy poleceń**, dzięki czemu możliwe jest tworzenie szybkich i wydajnych skryptów. Przykłady:

- tasklist | find "svchost.exe"
  (lista wykonywanych zadań zostanie przekazana do polecenia **find**)
- dir * | more
  (lista plików zostanie przekazana do polecania **more** – wyświetlającego tekst ekran po ekranie)
- attrib | sort
  (lista atrybutów zostanie przekazana do polecania **sort** – sortującego tekst)

**Zadanie 6.**

Zapoznać się z opisem komend i przełączników: **find**, **more** oraz **sort**.

**Zadanie 7.**

Wykorzystać przetwarzanie potokowe w celu:

- Wyświetlenia tylko adresów fizycznych z polecenia **ipconfig** –all
- Wyświetlenia tylko poleceń dotyczących plików (lista poleceń: **help**)
- Posortowania zawartości dowolnego pliku (wyświetlenie pliku: **type** plik)

## 6. Strumienie danych

W systemie Windows ekran monitora jest traktowany jako standardowe urządzenie wyjściowe.
Do urządzeń wejściowych możemy zaliczyć:

- klawiaturę - określaną jako konsola (CON).
- drukarkę (PRN),
- złącza szeregowe (COM1 do COM4),
- złącza równoległe (LPT1 do LPT3),
- urządzenie puste (NUL),
- synonim nazwy COM1 (AUX).

Przekierowania wykonywane są następująco:

- > - przekierowanie strumienia do wejścia
- >> - dopis strumienia danych do wejścia
- 2> - przekierowanie wyjścia błędów do wejścia
- 2>> - dopis wyjścia błędów do wejścia
- < - przkierowanie strumienia z wejścia

Przykłady:

- dir > lista.txt - zapisanie listy plików do pliku lista.txt
- mkdir test 2> NUL - błędy zostaną przekazane do urządzenia pustego (nie zostaną wyświetlone)

## 7. Zadania do wykonania

**Zadanie 8.**

- Zapisać zmienne środowiskowe do pliku zm1.txt
- Zapisać posortowane zmienne środowiskowe do pliku zm2.txt
- Porównać oba pliki za pomocą polecenia **fc**

**Zadanie 9.**

Napisać skrypt który wyszuka wszystkie pliki zawierające w nazwie "readme". Wynik skryptu zapisać do pliku readme.txt

**Zadanie 10.**

I. Wykorzystaj polecenie **tasklist** tak aby wyświetlało tylko procesy zajmujące więcej niż 100MB w pamięci (należy wykorzystać przełącznik /FI z odpowiednim filtrem).
II. Posortuj wynik powyższej komendy (polecenie **sort**) tak aby sortowało wyniki według rozmiaru zajmowanego w pamięci (sortowanie począwszy od znaku nr. **64**).
III. Napisz skrypt który będzie dopisywał wynik powyższego działania do pliku procesy%data%.txt (wartość %data% powinna zawierać bieżący dzień, miesiąc i rok).

**Zadanie 11*.** (rozwiązanie zadań oznaczonych * należy umieścić w sprawozdaniu lub pokazać podczas zajęć).

Napisz skrypt zapisujący statystyki połączeń użytkownika (net statistics workstation) do pliku C:\temp\net.txt. Do pliku należy dopisać tylko bieżącą datę, godzinę, oraz informacje o ilości danych nadanych i otrzymanych (w celu ich pozyskania należy wykorzystać komendę **find**). Skrypt powinien tworzyć katalog C:\temp\ (w przypadku gdy katalog istnieje błędy należy przekierować do strumienia pustego).

## 8. Zadania dodatkowe

**Zadanie 12.**

Napisz skrypt zliczający wystąpienie procesu zawierającego w nazwie "serv". Wykorzystaj tylko polecenie **fc**.

**Zadanie 13.**

Napisz skrypt umożliwiający kopiowanie plików o nazwach podanych przez użytkownika. Skrypt powinien też zapytać użytkownika jakiego polecenia kopiowania chce użyć (copy, xcopy lub robocopy).

**Zadanie 14.**

Zastanów się nad działaniem następującego skryptu:

```
sort < plik.txt | find "OK"
```

**Zadanie 15**.**

Napisz skrypt który będzie łączył w jeden plik wynikowy dowolną liczbę plików podanych jako argumenty skryptu. Nazwa pliku wynikowego powinna być pobrana od użytkownika podczas działania skryptu. Wykorzystaj polecenie **shift** oraz **goto**.

**Zadanie 16.**

Zapoznaj się z działaniem eksperymentalnego programu xBATED:

http://www.iisi.pcz.pl/index.php/pl/do-pobrania?func=fileinfo&id=1070