

Narzędzia informatyczne w językoznawstwie

Wiersz poleceń - Wprowadzenie

Marcin Junczys-Dowmunt

`junczys@amu.edu.pl`

Zakład Logiki Stosowanej

<http://www.logic.amu.edu.pl>

7. listopada 2007

- ▶ Wiersz poleceń (ang. Command Line Interface, CLI), to jeden z najczęściej spotykanych sposobów interakcji człowieka z komputerem.

- ▶ Wiersz poleceń (ang. Command Line Interface, CLI), to jeden z najczęściej spotykanych sposobów interakcji człowieka z komputerem.
- ▶ Inne przykłady to interfejs tekstowy oraz interfejs graficzny.

- ▶ Wiersz poleceń (ang. Command Line Interface, CLI), to jeden z najczęściej spotykanych sposobów interakcji człowieka z komputerem.
- ▶ Inne przykłady to interfejs tekstowy oraz interfejs graficzny.

Zasada działania

- ▶ Wydawania poleceń ze ściśle określonego zestawu i określonej składni
- ▶ Polecenia wpisywane z klawiatury lub zapisane w skryptach
- ▶ Przykłady: powłoki systemowe UNIX, Linux i DOS lub np. gnuplot

Ogólna składnia poleceń

Polecenia dla komputera wydawane w wierszu poleceń mają z reguły następującą postać:

```
zrób_coś w_sposób w_stosunku_do_czegoś
```

Ogólna składnia poleceń

Polecenia dla komputera wydawane w wierszu poleceń mają z reguły następującą postać:

```
zrób_coś w_sposób w_stosunku_do_czegoś
```

lub

```
zrób_coś w_sposób < plik_wejściowy > plik_wyjściowy
```

Ogólna składnia poleceń

Polecenia dla komputera wydawane w wierszu poleceń mają z reguły następującą postać:

```
zrób_coś w_sposób w_stosunku_do_czegoś
```

lub

```
zrób_coś w_sposób < plik_wejściowy > plik_wyjściowy
```

lub

```
zrób_coś w_sposób < plik_wejściowy | zrób_coś >  
plik_wyjściowy
```

Ogólna składnia poleceń

Polecenia dla komputera wydawane w wierszu poleceń mają z reguły następującą postać:

```
zrób_coś w_sposób w_stosunku_do_czegoś
```

lub

```
zrób_coś w_sposób < plik_wejściowy > plik_wyjściowy
```

lub

```
zrób_coś w_sposób < plik_wejściowy | zrób_coś >  
plik_wyjściowy
```

Czyli polecenia mają zwykle postać podobną do czasowników (nazwa polecenia) z okolicznikami (opcje) i dopełnieniami (pliki) lub zdaniami podrzędnymi (potoki)

Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

- ▶ Można niektóre czynności wykonać szybciej (np. `del *.txt`)

Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

- ▶ Można niektóre czynności wykonać szybciej (np. `del *.txt`)
- ▶ Możemy np. zapisać informacje wyjściowe danej komendy do pliku tekstowego w celu późniejszego przetwarzania (np. `dir *.txt >names.dat`)

Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

- ▶ Można niektóre czynności wykonać szybciej (np. `del *.txt`)
- ▶ Możemy np. zapisać informacje wyjściowe danej komendy do pliku tekstowego w celu późniejszego przetwarzania (np. `dir *.txt >names.dat`)
- ▶ Można zautomatyzować powtarzające się czynności (np. `for %f in (*.txt) do sort %f >%f.sorted`) i tworzyć pliki wsadowe (skrypty)

Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

- ▶ Można niektóre czynności wykonać szybciej (np. `del *.txt`)
- ▶ Możemy np. zapisać informacje wyjściowe danej komendy do pliku tekstowego w celu późniejszego przetwarzania (np. `dir *.txt >names.dat`)
- ▶ Można zautomatyzować powtarzające się czynności (np. `for %f in (*.txt) do sort %f >%f.sorted`) i tworzyć pliki wsadowe (skrypty)
- ▶ Możemy używać programy, które nie mają interfejsów graficznych, np. małe skrypty w PERL.



Po co komu wiersz poleceń (konsola) ...

... skoro możemy to samo osiągnąć za pomocą pulpitu graficznego i myszki?

- ▶ Można niektóre czynności wykonać szybciej (np. `del *.txt`)
- ▶ Możemy np. zapisać informacje wyjściowe danej komendy do pliku tekstowego w celu późniejszego przetwarzania (np. `dir *.txt >names.dat`)
- ▶ Można zautomatyzować powtarzające się czynności (np. `for %f in (*.txt) do sort %f >%f.sorted`) i tworzyć pliki wsadowe (skrypty)
- ▶ Możemy używać programy, które nie mają interfejsów graficznych, np. małe skrypty w PERL.
- ▶ Możemy korzystać ze potoków wejścia/wyjścia do tworzenia łańcuchów programów (np. `type *.txt | sort | more`)



Otwieramy okno konsoli!

Na Windows XP można to na kilka sposobów, zwykle to:

- ▶ Start → Uruchom → cmd 
- ▶ Start → Wszystkie Programy → Akcesoria → Wiersz poleceń 

Otwieramy okno konsoli!

Na Windows XP można to na kilka sposobów, zwykle to:

- ▶ Start → Uruchom → cmd 
- ▶ Start → Wszystkie Programy → Akcesoria → Wiersz polecenia 

Widzimy raczej prosty interfejs

- ▶ *Prompt* (standardowo: C:\Document and Settings\imię>) wskazuje na aktualny folder w którym się znajdujemy
- ▶ Wpisujemy tu jakieś polecenie i potwierdzamy klawiszem *Enter*
- ▶ Polecenie zostanie wykonane, jego wyjście zostanie wyświetlone w konsoli
- ▶ Pojawia się kolejny prompt itd.

- ▶ Wypisujemy `dir` i potwierdzamy klawiszem Enter.
- ▶ Polecenie wyświetli listę plików i katalogów zawartych w aktualnym katalogu

Pytanie

Ile plików i ile katalogów wyświetliło to polecenie?

- ▶ Wypisujemy `dir` i potwierdzamy klawiszem Enter.
- ▶ Polecenie wyświetli listę plików i katalogów zawartych w aktualnym katalogu

Pytanie

Ile plików i ile katalogów wyświetliło to polecenie?

- ▶ Wypisujemy `cd \` (cd backslash) i potwierdzamy klawiszem Enter.

Pytania

Jak teraz wygląda prompt i co to oznacza?

Jakie pliki znajdują się w katalogu głównym?

Poruszanie się po katalogach

Proszę wpisać:

1. `cd C:\Document and Settings\Student\Pulpit`
2. `dir`
3. `mkdir Konsola`
4. `dir`
5. `cd Konsola`
6. `dir`
7. `cd ..`
8. `dir`
9. `cd Konsola`

Pytania

Co dzieje się w każdym kroku?

Czemu służy polecenie `mkdir Konsola`, a czemu `cd ..` ?

Poznaliśmy właśnie kilka najbardziej podstawowych poleceń

- ▶ Wpisując cd plus ścieżka zmieniamy bieżący katalog.
Ścieżka może być bezwzględna (1.) lub względna (5. i 9.)

Poznaliśmy właśnie kilka najbardziej podstawowych poleceń

- ▶ Wpisując cd plus ścieżka zmieniamy bieżący katalog. Ścieżka może być bezwzględna (1.) lub względna (5. i 9.)
- ▶ Istnieją dwie specjalne względne ścieżki, mianowicie .. (7.) oraz .
 - .. oznacza *katalog nadrzędny*
 - . oznacza *katalog bieżący*

Poznaliśmy właśnie kilka najbardziej podstawowych poleceń

- ▶ Wpisując `cd` plus ścieżka zmieniamy bieżący katalog. Ścieżka może być bezwzględna (1.) lub względna (5. i 9.)
- ▶ Istnieją dwie specjalne względne ścieżki, mianowicie `..` (7.) oraz `.`
 - `..` oznacza *katalog nadrzędny*
 - `.` oznacza *katalog bieżący*
- ▶ Polecenie `dir` wyświetla zawartość bieżącego katalogu lub katalogu podanego w ścieżce np. `dir \ "Program Files"`

Poznaliśmy właśnie kilka najbardziej podstawowych poleceń

- ▶ Wpisując cd plus ścieżka zmieniamy bieżący katalog. Ścieżka może być bezwzględna (1.) lub względna (5. i 9.)
- ▶ Istnieją dwie specjalne względne ścieżki, mianowicie .. (7.) oraz .
 - .. oznacza *katalog nadrzędny*
 - . oznacza *katalog bieżący*
- ▶ Polecenie dir wyświetla zawartość bieżącego katalogu lub katalogu podanego w ścieżce np. dir \"Program Files\"

Pytanie

Wpisać explorer . – Co się stało?

Niektórzy osoby nie lubią konsoli, bo wpisywanie powtarzających się poleceń jest niewygodne.

Niektórzy osoby nie lubią konsoli, bo wpisywanie powtarzających się poleceń jest niewygodne.

- ▶ Wciskamy strzałki ↑ lub ↓ żeby przywołać wcześniej wpisane polecenia.

Pytanie

Która komenda pojawia się po pierwszym naciśnięciu ↑, a która po kolejnym?

Niektórzy osoby nie lubią konsoli, bo wpisywanie powtarzających się poleceń jest niewygodne.

- ▶ Wciskamy strzałki ↑ lub ↓ żeby przywołać wcześniej wpisane polecenia.

Pytanie

Która komenda pojawia się po pierwszym naciśnięciu ↑, a która po kolejnym?

- ▶ Wpisujemy c i następnie wciskamy klawisz F8

Pytanie

Która komenda pojawia się po pierwszym naciśnięciu F8, a która po kolejnym? Co się dzieje?

Ćwiczenie

- ▶ Wpisać cd \Windows (dokładnie tak!)

Po otrzymaniu komunikatu o błędzie wykorzystać przedstawione metody by możliwe szybko poprawić błąd. Jakie czynności zostały wykonane?

Ćwiczenie

- ▶ Wpisać cd \Windows (dokładnie tak!)

Po otrzymaniu komunikatu o błędzie wykorzystać przedstawione metody by możliwe szybko poprawić błąd. Jakie czynności zostały wykonane?

Ćwiczenie

Wpisać cd \W i nacisnąć klawisz \leftrightarrow (Tab) — Co się stało?

Wpisać cd Windows\ i nacisnąć wielokrotnie klawisz \leftrightarrow (Tab) —
Co się dzieje?

Funkcja ta nazywa się *automatycznym uzupełnianiem nazw plików* (filename autocompletion)

Wpisujemy:

- ▶ `cd \Document and Settings\Student\Pulpit\Konsola`
- ▶ `dir`
- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest drugi plik >plik2.txt`
- ▶ `dir`

Pytanie

Co się zmieniło między pierwszym `dir` a drugim `dir`?

Do tworzenia plików wykorzystujemy potoki, ale o tym więcej na następnych zajęciach .

Pytanie

Co się dzieje po każdym z następujących poleceń?

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1 (lub mkdir katalog1)`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1 (lub mkdir katalog1)`
- ▶ `md katalog2`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1 (lub mkdir katalog1)`
- ▶ `md katalog2`
- ▶ `copy tralala.txt katalog1\`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1 (lub mkdir katalog1)`
- ▶ `md katalog2`
- ▶ `copy tralala.txt katalog1\`
- ▶ `del katalog1`

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1` (lub `mkdir katalog1`)
- ▶ `md katalog2`
- ▶ `copy tralala.txt katalog1\`
- ▶ `del katalog1`
- ▶ `rd katalog1` (lub `rmdir katalog1`)

Pytanie

Co się dzieje po każdym z następujących poleceń?

- ▶ `del plik1.txt`
- ▶ `copy plik2.txt plik1.txt`
- ▶ `rename plik1.txt tralala.txt`
- ▶ `md katalog1` (lub `mkdir katalog1`)
- ▶ `md katalog2`
- ▶ `copy tralala.txt katalog1\`
- ▶ `del katalog1`
- ▶ `rd katalog1` (lub `rmdir katalog1`)
- ▶ `move plik2.txt katalog2`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

▶ echo To jest pierwszy plik >plik1.txt

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ echo To jest pierwszy plik >plik1.txt
- ▶ echo To jest pierwszy plik >plik2.txt

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`
- ▶ `copy pli??.* tes??.*`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`
- ▶ `copy pli??.* tes??.*`
- ▶ `del *.txt`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`
- ▶ `copy pli??.* tes??.*`
- ▶ `del *.txt`
- ▶ `mkdir katalog1`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`
- ▶ `copy pli??.* tes??.*`
- ▶ `del *.txt`
- ▶ `mkdir katalog1`
- ▶ `move *.* katalog1`

Wieloznaczniki: * (ogólny) i ? (lokalny)

Jeśli chcemy skopiować wszystkie pliki danego typu warto skorzystać z tzw. *wieloznaczników*, czyli znaku *

Znak * jest substytutem dowolnego ciągu znaków w nazwach plików.

- ▶ `echo To jest pierwszy plik >plik1.txt`
- ▶ `echo To jest pierwszy plik >plik2.txt`
- ▶ `copy *.txt *.dat`
- ▶ `copy pli??.* tes??.*`
- ▶ `del *.txt`
- ▶ `mkdir katalog1`
- ▶ `move *.* katalog1`

Pytanie

Jak uzyskać podobne efekty bez pomocy konsoli?

Większość poleceń posiada dodatkowe opcje, które modyfikują sposób działania poleceń.

Prawie każde wbudowane polecenie konsoli posiada opcję /?, która wyświetla informacje o pozostałych opcjach.

Większość poleceń posiada dodatkowe opcje, które modyfikują sposób działania poleceń.

Prawie każde wbudowane polecenie konsoli posiada opcję /?, która wyświetla informacje o pozostałych opcjach.

np.

- ▶ `dir /?`
- ▶ `dir /w`
- ▶ `dir /o-n /w`

Większość poleceń posiada dodatkowe opcje, które modyfikują sposób działania poleceń.

Prawie każde wbudowane polecenie konsoli posiada opcję /?, która wyświetla informacje o pozostałych opcjach.

np.

- ▶ `dir /?`
- ▶ `dir /w`
- ▶ `dir /o-n /w`

Uwaga

Sposób zapisu opcji w konsoli Windows nie jest do końca standardowy porównując z innymi systemami np. Linuxem, gdzie opcje są zapisywane w np. `ls -la`

Polecenia działające na plikach tekstowych

- ▶ echo – Wyświetla komunikat wpisany jako argument
np. echo To jest komunikat

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`
- ▶ `more` – Wyświetla dane po jednym ekranie na raz
np. `more plik1.txt`

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`
- ▶ `more` – Wyświetla dane po jednym ekranie na raz
np. `more plik1.txt`
- ▶ `sort` – sortuje wiersze w podanym pliku i wyświetla na konsoli
np. `sort plik1.txt`

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`
- ▶ `more` – Wyświetla dane po jednym ekranie na raz
np. `more plik1.txt`
- ▶ `sort` – sortuje wiersze w podanym pliku i wyświetla na konsoli
np. `sort plik1.txt`
- ▶ `fc` – porównuje dwa pliki i wyświetla różnice między nimi
np. `fc plik1.txt plik2.txt`

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`
- ▶ `more` – Wyświetla dane po jednym ekranie na raz
np. `more plik1.txt`
- ▶ `sort` – sortuje wiersze w podanym pliku i wyświetla na konsoli
np. `sort plik1.txt`
- ▶ `fc` – porównuje dwa pliki i wyświetla różnice między nimi
np. `fc plik1.txt plik2.txt`
- ▶ `find` – szuka ciągu znaków w pliku lub wielu plikach
np. `find /N "ciąg" *.txt`

Polecenia działające na plikach tekstowych

- ▶ `echo` – Wyświetla komunikat wpisany jako argument
np. `echo To jest komunikat`
- ▶ `type` – Wyświetla zawartość jednego lub wielu plików
np. `type *.txt`
- ▶ `more` – Wyświetla dane po jednym ekranie na raz
np. `more plik1.txt`
- ▶ `sort` – sortuje wiersze w podanym pliku i wyświetla na konsoli
np. `sort plik1.txt`
- ▶ `fc` – porównuje dwa pliki i wyświetla różnice między nimi
np. `fc plik1.txt plik2.txt`
- ▶ `find` – szuka ciągu znaków w pliku lub wielu plikach
np. `find /N "ciąg" *.txt`
- ▶ `chcp` – Ustawia stronę kodową konsoli
np. `chcp 65001` (ustawia kodowanie konsoli na utf-8)

Następnym razem

- ▶ Potoki i filtry
- ▶ Pliki wsadowe, czyli automatyzacja