ОСНОВИ НА ИЗПОЛЗВАНЕТО НА КОМАНДНИЯ ИНТЕРПРЕТАТОР НА MS WINDOWS

Изпълняване на команди

Изпълняването на команди в операционната система MS Windows може да става по два начина. Основният е да се стартира командният интерпретатор и командите да се въвеждат в него, обикновено по една на ред. Друг начин е командите да се запишат във файл с име, завършващо на .bat. Такъв файл се нарича команден или командна процедура и представлява изпълнима програма. Както всяка друга програма, той може да бъде изпълнен – което в случая означава да се изпълнят записаните в него команди – чрез щракане с мишката върху името или представящия го знак – или в командния интерпретатор като команда с име – името на файла.

Тук не разглеждаме как се съставят и изпълняват командни процедури, нито множеството команди или командния език на MS Windows като цяло. Представяме само неголям брой команди за работа с файлове и директории и необходимите за разбирането на тези команди общи положения. Познаването на общите положения е полезно и за грамотното изпълняване на конзолни програми по различни начини, включително съчетавайки ги една с друга.

Команден интерпретатор, конзола и конзолен прозорец

За стартиране на командния интерпретатор трябва да се напише cmd (и да се натисне клавиша Enter) в реда за търсене на стартовото меню на системата. При това действие се отваря **конзолен прозорец** с подсказка за въвеждане на команди.

Конзолата е понятие, обединяващо виртуална клавиатура и виртуален екран, които позволяват да въвеждаме данни за програма или команда и да извеждаме резултатите от изпълнението ѝ. Затова от гледна точка на програмата разглеждаме виртуалната клавиатура като **стандартен вход**, а виртуалния екран – като **стандартен изход**. Обикновено ролята на виртуална клавиатура играе действителната клавиатура на компютъра, а ролята на виртуален екран – частта от екрана на компютъра, заемана от конзолния прозорец, но всяко от двете може да се променя чрез **пренасочване**. За що и как пренасочването може да бъде удобно ще разгледаме по-нататък.

В обичайния режим на работа с командния интерпретатор потребителят изпълнява команди една след друга, като написва името и съответните аргументи на всяка. По същия начин се изпълняват и програми, създадени от самия потребител или от някого другиго. Командният интерпретатор не прави разлика между командите на операционната система и приложните програми – за него всички те са команди, които биват изпълнявани, като ги цитираме по име.

Програма, която не използва графичния интерфейс на операционната система се нарича конзолна програма. Всяко въвеждане и извеждане на информация при такава програма става чрез нейната конзола, файлове или Интернет, като в много случаи е само първото. Ако при това стандартните вход и изход не са пренасочени, въвеждането и извеждането на информация се реализират чрез конзолния прозорец, в който се изпълнява програмата, и са непосредствено видими на екрана като текст: конзолният прозорец е програмна реализация на понятието конзола. Поради това и за краткост често с думата „конзола“ наричаме и кой да е конзолен прозорец.

Файлове, директории и цитирането им

Изпълняването на редица команди изисква посочване на файлове, а за това е важно да се знае къде именно се намират файловете. Файловете се разполагат в **директории** върху различни устройства – най-често твърди или други дискове. Върху всяко устройство директориите и файловете образуват йерархично организирана структура: **файловата система на устройството**. Коренът на тази система е **главната директория на устройството**. В нея има файлове и други директории, в тези директории също може да има файлове и директории и т.н.

В имената на файлове и директории освен букви могат да участват различни други знаци, но има някои ограничения, например не може да се среща никое от / \ < > | ? * ". Знакът . (точка) може да бъде част от име, но той се отличава със следното: частта след . се смята за разширение на името, а . и .. сами по себе си са особени имена, чийто смисъл изясняваме по-нататък.

Ако програмата се нарича prog и входът ѝ трябва да идва от файла in.txt, пускането на програмата би имало следния вид:

```
prog <in.txt
```

(между < и името на файла може да има интервал, това е без значение).

За пренасочване на стандартния изход се използват знаците > или >>. При > се образува нов изходен файл, дори ако файл с посоченото име е съществувал преди пускането на програмата (и тогава текущото му съдържание се унищожава). При >> съдържанието на файла, който става изходен не се унищожава, а само се допълва от края нататък от това, което извежда програмата. Пускането на програмата prog с пренасочване на изхода към файл out.txt изглежда така:

```
prog >out.txt
```

Възможно е също да пуснем програмата, пренасочвайки и входа, и изхода:

```
prog <in.txt >out.txt
```

Всяка програма може да бъде написана така, че да проверява наличието на данни на стандартния си вход: това е необходимо, когато програмата трябва да въведе неопределено количество данни – колкото бъдат подадени при всяко конкретно пускане. Когато входът е пренасочен към файл обемът му се определя от съдържанието на файла, така че няма нужда изчерпването да се задава с явно действие. Когато стандартният вход е въвежданото от клавиатурата, потребителят обозначава края на въвеждането (изчерпването на входа) с натискане на нов ред на клавиша Ctrl и после, докато е натиснат Ctrl, на клавиша z – при това на екрана се изписва ^Z. След това се натиска Enter, както при завършване на ред.

Особена форма на пренасочване е **конвейерът**, наричан също **канал** – чрез него стандартният изход на дадена програма става стандартен вход на друга. За образуване на конвейер между имената на програмите се записва |, както например в

```
prog1 | prog2
```

където изходът на prog1 постъпва непосредствено като вход на prog2. При такова пускане prog1 и prog2 фактически стават една нова програма, стандартният вход на която е този на prog1, а стандартният изход – този на prog2, при което тези останали вход и изход в горното пускане не са пренасочени. Когато е нужно и това, то става както за самостоятелна програма. Например, за да пренасочим и входа, и изхода на конвейера prog1|prog2 записваме

```
(prog1 | prog2) <in.txt >out.txt
```

Конвейер може да се образува и от повече от две програми – записването на такова действие е очевидно обобщение на горното.

От гледна точка на възможността за пренасочване програма, която получава и извежда данни съответно от стандартния вход и на стандартния изход се нарича **филтър**: тя може да се разглежда като преобразувател на текст, който може да постъпва от файл или друга програма и да бъде насочен към файл или програма. Свързвани в конвейер, всеки две или повече програми филтри образуват също филтър – получаваме филтри едни от други посредством композиране.

Пренасочването е полезно в различни случаи. Например при тестване на програма, когато тя се пуска многократно с едно и също или с няколко множества от входни данни е удобно да запишем тези данни в съответния брой файлове и при пускането на програмата да пренасочваме входа към който е нужно от тях. С това се избягват и многократното въвеждане на едно и също, и възможните при такова повтаряне грешки, а заедно с това се осигурява въвкавост – избирането на конкретен вариант на входни данни е възможно най-лесно, изисквайки само незначителна промяна при пускането на програмата.

Не по-малко полезна е възможността да пренасочваме изхода. Такова пренасочване е много удобно например за записване на резултатите от програмата при различни нейни пускания, независимо от това, че в самата програма такава възможност не е предвидена. Това е толкова по-съществено, когато резултатът е с голям обем и за да го разглеждаме или търсим конкретни стойности в него е необходимо да прибягваме до текстов редактор или друга програма.

Конвейерът е много удобен, когато изходът на дадена програма трябва да бъде обработен от друга. Такъв е случаят например когато изходът бива проверяван за правилност, в него търсим някакви свойства или извличаме отговарящи на някакво условие части.

Като цяло механизмът на пренасочването е сам по себе си важно средство за програмиране чрез използване на готови програми – чрез него можем да образуваме всякакви нови програми просто като свързваме в едно наличните.

Някои команди

Повечето команди, освен аргументи от общ вид, които най-често са имена на файлове и директории, могат да имат и ключове. Ключовете служат за уточняване на действието на командите. Всеки ключ се задава чрез буква, предшествана от знака / и може да изисква свои аргументи.

`c:  d:  e: ...`
: Това е множество от команди с имена, съвпадащи с тези на устройствата. След изпълняването на такава команда текущо за командния интерпретатор става посоченото устройство, а текуща директория – текущата за това устройство директория.

`cd` *директория*
: Смяна на текущата директория на командния интерпретатор или устройство. Примери:

`cd ..`	иди една стъпка по-нагоре (в родителската директория);
`cd ..\..`	иди две стъпки по-нагоре;
`cd \`	иди в главната на текущото устройство директория;
`cd c:tmp`	смени текущата директория на устройство c: с вложената в нея tmp;
`cd c:..\..`	смени текущата директория на устройство c: с директорията на две стъпки по-нагоре.

  В двата последни случая се сменя текущата директория на устройство c:. При това текущата директория на командния интерпретатор може да не се смени – това става само когато устройството c: е текущо (дали е така се вижда в подсказката на командния интерпретатор). Ако трябва с една команда и да сменим текущата директория на дадено устройство, и да направим това устройство текущо (с което и избираме текущата на него директория за текуща на командния интерпретатор) можем да използваме ключа /d, както например в командата `cd /d c:..\...`

`dir` *директория*
: Показване на съдържанието на директория – посочена чрез аргумент или текущата, ако аргумент няма. С ключа /s нареждаме да се показва и съдържанието на поддиректориите. Със задаване на ключа /p изходът се оформя постранично, все едно че минава през командата more (виж по-долу). С ключа /o и буква след него се избира подреждане на изхода от dir по признак както следва:
  - n азбучен ред на имената;
  - s нарастване на размера на файловете;
  - e азбучен ред на разширенията;
  - d нарастване на времето на последна промяна;
  - g първо поддиректориите.

  Ако буквата се предшества от знака -, прилага се обратното подреждане.

`copy` *кое къде*
: Копиране на файл или файлове. Аргументът *кое* посочва име на файл или шаблон за множество от имена на файлове, които трябва да се копират. Ако аргументът *къде* е име на директория, файловете се копират в нея, а ако е име на файл, от съдържанието на зададените чрез *кое* файл или файлове заедно се образува копие в единствен файл с посоченото от *къде* име. Множество файлове в *кое* се задава или като използваме шаблон, или като зададем *кое* във вида *файл+файл+⋯+файл*.

Примери:

`copy p.txt q.txt`	файлът p.txt се копира в q.txt;
`copy p.txt q.txt >nul`	както по-горе, но без съобщение след завършване;
`copy p.txt xy\`	създава се копие на p.txt във вложената директория xy;
`copy ..\p.txt .`	файлът p.txt от родителската директория се копира в текущата;
`copy *.dat ..`	всички файлове с разширение dat се копират в родителската директория;
`copy *.dat ..\a.dat`	в родителската директория се създава файл a.dat, съединявайки всички файлове с разширение dat от текущата директория;
`copy a.h + b.h n.txt`	създава се файл n.txt, съединявайки a.h и b.h;
`copy con x`	текстът от стандартния вход се записва във файла x;
`copy x con`	съдържанието на файла x се извежда на стандартния изход (както при type – виж по-долу).

Предпоследният пример показва как да създадем файл от какъв да е брой редове, които въвеждаме непосредствено от клавиатурата. Ако въведем текст без да натискаме клавиша Enter, а после Ctrl-Z и едва тогава Enter, във файла ще бъде записан ред без знака за край на ред. Ако използваме ключа /-y след думата copy, ще накараме интерпретатора да пита за потвърждаване на промяната на всеки файл в *къде*, който съществува отпреди изпълняването на командата. Ключът /y позволява файловете да се променят без питане за потвърждаване.

`xcopy` *кое къде*

Разширен вариант на команда за копиране на файлове. В частност позволява копиране на цели директории заедно със съдържанието им. Тази команда има много ключове. Ключът /e води до копиране на цялото съдържание на директориите, включително поддиректориите с тяхното съдържание. Ключът /s има подобно действие, но при него празните поддиректории не се копират. Ако е зададен ключа /i и се копира повече от един файл, а името *къде* не отговаря на съществуващ файл или директория, смята се, че това трябва да е директория и тя се създава.

`move` *кое къде*

Преместване на файл или файлове. Аргументът *къде* трябва явно или неявно (както например ако е c:) да задава име на директория. Ако и *кое* е име на директория, тази директория или се премества, или се преименува според това дали *къде* отговаря на съществуваща директория или на ново име в първата. (Във втория случай същото се получава с командата ren.) Както при copy и xcopy, и при move може да се използва ключ /-y или /y.

`ren` *име1 име2*

Преименуване на файла *име1* в *име2*. Когато *име1* е шаблон, могат да се преименуват множество файлове. Аргументът *име2* е или име, или шаблон без име на устройство и имена на директории. Например ren *.cxx *.cpp променя в текущата директория разширенията cxx на имена на файлове в cpp.

`del` *име име* ...

Унищожаване на файловете с посочените имена. Вместо кое да е от имената може да бъде посочен шаблон. Ако е зададен ключ /p командният интерпретатор ще изисква потвърждаване на действието за всеки файл. С ключа /s, ако дадено име отговаря на директория, унищожават се съдържащите се в нея файлове.

`md` *име*

Създаване на директория с посоченото име в текущата директория или в текущата директория на посоченото заедно с името устройство. Например md c:abc създава директория abc в текущата директория на устройство c:.

Когато име на устройство се използва самостоятелно на място, където се предполага да се посочва директория, подразбира се текущата директория на цитираното устройство. В по-общия случай името на устройство се използва съвместно с пълното или относително име на директория или файл. Например d:\ именува главната директория на устройството d:, d:\pgm\re е име на файл или директория re в директорията pgm, намираща се в главната директория на устройството d:, а d:pgm\re\ е име на директория re в директориятa pgm, намираща се в текущата на устройството d: директория. (Обръщаме внимание на ролята на \ в последните два примера.)

Шаблони за имена на файлове

Понякога е желателно дадена команда на операционната система да се отнася до множество файлове, включително неопределено много на брой. Ясно е, че в такъв случай е нужно цялото множество да бъде представено с единствен аргумент. Това става, като вместо име на файл използваме образец, шаблон. Шаблонът е обобщено име, на което посредством прости правила съответства множество от действителни имена.

Образуването на шаблон става с помощта на знаците ? и *, които именно поради тази си роля не могат да участват в имена на файлове и директории.

Знакът ? в шаблон означава, че на даденото място в съответстващо на шаблона име на файл може да стои който и да е знак или изобщо да няма знак. Например, ако в шаблона се среща p?xy, на тази част на шаблона отговаря всяка част от име, започващо с p и завършващо с xy, като между тях може да има още един, какъвто и да е знак.

Знакът * в шаблон задава присъствие в съответстващо на шаблона име на какъв да е, възможно и празен, низ от знаци. Например на p*xy, съответства част от име, започващо с p и завършващо с xy, като между тях може да има колкото и да е и какви то и да е знаци.

По-долу са дадени няколко примера на шаблони и съответстващи им имена. Напомняме, че частта от името след знака . (точка) се нарича разширение на името.

`?.??`	Празно или еднознаково име с празно или състоящо се от един или два знака разширение.
`?.*`	Празно или еднознаково име с празно или непразно разширение.
`*.`	Непразно име без разширение.
`*.txt`	Име с разширение txt.
`s*.cpp`	Започващо с s име с разширение cpp.
`hd*-*.cpp`	Започващо с hd и съдържащо - (минус) име с разширение cpp.

Трябва да се има предвид, че когато разширение на името няма, отсъства и отделящата го от името точка. А когато шаблонът допуска да няма и име, и разширение, може наистина да липсва само едното от тях.

Пренасочване

Редица команди, а и изобщо много програми вземат данните си от стандартния вход и извеждат резултатите си на стандартния изход. Стандартният вход по начало е това, което въвеждаме от клавиатурата, но понякога е полезно да накараме програмата да приема данни не от клавиатурата, а от един или друг файл. Това става посредством пренасочване. Пренасочването на стандартния вход е действие на операционната система, по-точно на командния интерпретатор, при което „стандартен вход“ на дадена програма става съдържанието на посочен файл. Самата програма не се променя: тя е съставена така, че да въвежда данни именно от стандартния вход и нито „знае“, нито би могла да знае дали при конкретното ѝ изпълнение той е въвеждан от клавиатурата или е налице пренасочване.

Аналогично възможно е при конкретно пускане на програма стандартният изход да бъде пренасочен, така че програмата да извежда резултатите си във файл вместо на екрана. И в този случай програмата не различава какво конкретно е стандартният изход в случая – екранът или файл. Това зависи от начина, по който пускаме програмата в командния интерпретатор и при различни пускания може да е различно.

За да бъде пренасочен стандартният вход, след името на програмата и, ако има такива, аргументите на повикването ѝ, поставяме знак <, следван от името на файл.

Имената con, nul, prn, aux, а също имената от com1 до com9 и от lpt1 до lpt9 не могат да именуват файлове и директории нито сами по себе си, нито с разширения. Това са имена на псевдофайлове и някои устройства. Например името con обозначава стандартния вход или стандартния изход според това как е използвано. Името nul отговаря на условно устройство, в което, каквото и да запишем, „изчезва“ – всъщност просто не се записва.

Впрочем, допустими имена за файлове и директории са .nul, .con, con0, com0 и други такива.

Ясно е, че възможността да записваме файлове в една или друга директория на едно или друго устройство усложнява или дори прави невъзможно еднозначното различаване на файл само по име, още повече, че едно и също име на файл може да се среща в различни директории и навсякъде то обозначава различни файлове. Еднозначното посочване на файл изисква задаване на неговото **пълно име**. Пълното име се състои от име на устройство, пътя до файла и собственото име на файла. **Пътят** е списък от имена на директории, започващ от главната за устройството и отразяващ влаганията на директории една в друга, които следваме до достигане на тази, която непосредствено съдържа файла. Например името c:\xy\pqr\abc.txt е пълното име на файла със собствено име abc.txt, намиращ се на устройството c: в директорията \xy\pqr. Поточно, файлът се намира в директорията pqr, вложена в директорията xy, която пък се намира в главната.

Същото важи и за имената на директории. В горния пример c:\xy и c:\xy\pqr са пълните имена на директории. Същите имена можем да запишем във вида c:\xy\ и c:\xy\pqr\.

В името на файл или директория може да има интервали (шпации). Интервалите служат и за разделяне на името на коя да е команда от аргументите ѝ, както и на аргументите помежду им. Ако аргумент е име, съдържащо интервали, командният интерпретатор би го взел за два или повече отделни аргумента. За да се избегне това, записваме името между двойни кавички, както например при някои служебни за Windows директории – "c:\Program Files" и пр.

Текущо местоположение. Относителни имена

Задаването на пълните имена на файлове и директории може да е затруднително и да бави. Удобно е то да може да се избягва без да губим еднозначността на цитирането. Такава възможност е налице благодарение на уславянията за **текущо устройство** и **текуща директория**. Ако файл бива цитиран само чрез собствено име, подразбира се, че файлът се намира на устройството и в директорията на това устройство, които са текущи, когато се изпълнява цитиращата файла команда. Същото важи и за имената на директории. За удобство подсказката на командния интерпретатор обикновено съдържа имената на текущото устройство и на текущата директория на него. Подразбирането за текущи устройство и директория включва и това, че текущата директория на текущото устройство се смята и изобщо за текуща при изпълняване на команди.

Удобна възможност са и **относителните имена**. Тълкуването на такива имена е спрямо текущата директория. Ако текущата за командния интерпретатор директория е c:\xy, името pqr се разбира като пълното име c:\xy\pqr, а pqr\abc.txt – като c:\xy\pqr\abc.txt. Кое да е собствено име като pqr в примера е в този смисъл частен случай на относително име. Затова, ако текуща директория е c:\xy\pqr, името abc.txt отговаря на пълното c:\xy\pqr\abc.txt.

Във всяка директория присъства особеното име . или все едно .\ – то е относително име на директорията, когато тя е текуща. Във всяка освен главната директория се среща и името .. или все едно ..\ – то обозначава „родителската“ директория – тази, в която се намира дадената. Така, ако текуща директория е c:\xy\pqr, относителното име .. отговаря на пълното c:\xy, а ..\uvw – на името c:\xy\uvw. Името ..\..\ пък тогава посочва c:\ – главната директория на устройството c:.

Да отбележим, че имената на съдържащи файлове устройства като дискове, дялове на даден диск, устройства за CD, DVD или Blu-ray и флашки се състоят от по една буква на латиницата с двоеточие: c:, d:, e: и т.н. (Някога използваните имена a: и b: обозначаваха устройства за дискети и днес не се употребяват.)

rd *име*

Унищожаване на посочената директория. Ако името задава устройство, има се предвид директория на това устройство. Ако е зададен ключ /s и ако посочената директория е непразна, унищожават се файловете и поддиректориите в нея заедно със самата директория. (Без /s, ако директорията е непразна, командата се смята за неправилна и не се изпълнява.)

type *име*

Извеждане на съдържанието на посочения файл на стандартния изход. Чрез пренасочване се получава копиране на файл, при това без съпровождащо съобщение.

Примери:

type pgm.cpp	извежда се съдържанието на файла pgm.cpp;
type p.cpp >p.txt	съдържанието на файла p.cpp се копира в p.txt;
type nul >tmp	съдържанието на файла tmp се изтрива или се създава празен файл tmp;
type nul >>tmp	създава се празен файл tmp само ако такъв файл не съществува.

echo *текст*

Извеждане на *текст* на стандартния изход. Използва се най-вече за създаване на файл с посоченото съдържание или за добавяне на текст към файл – с пренасочване съответно > или >>.

Ако в извеждания от командата echo текст има особени за командния интерпретатор знаци, тяхната особеност трябва да се потисне, като запишем пред всеки такъв знак ^. Това се отнася и за самия знак ^, както се вижда в следните примери:

echo p -^> q	извежда се p → q;
echo a ^< b ^| a ^> b	извежда се a < b | a > b;
echo p ^^ q	извежда се p ^ q.

За да запишем празен ред във файл, например x, можем да изпълним

echo:>x или echo:>>x

(пред > не трябва да има интервал, иначе той ще присъства и във файла, т.е. извежданият ред няма да бъде съвсем празен).

more *име*

Извеждане на екрана на съдържанието на файл или, ако *име* не е посочено, на стандартния вход. Във втория случай, като правило, входът на тази команда се получава по конвейер, както например при dir|more или type t.txt|more. Ако изведданият текст е достатъчно голям, отначало се извежда толкова от него, че да напълни екрана. С натискане на клавиша Enter или ␣ (интервал) се извежда съответно следващият ред или следващата страница (екран) и т.н. до изчерпване, а с натискане на q или Ctrl-C извеждането се изоставя и командата завършва. Ако е зададен ключ /c, преди началото на извеждането съдържанието на екрана се изтрива.

cls

Изтриване на съдържанието на екрана.

exit

Завършване на работата на командния интерпретатор (и затваряне на конзолния прозорец).

Бойко Банчев, юни 2022