

# **Souhrnné informace o aplikaci TCPBatch**

Copyright: © 1997 ELEPS software

Verze: **1.10 / shareware / 13. července 1997**

Autor: **RNDr. František Adamec**

Vytvořeno za použití Microsoft™ Visual C++ ver. 4.1 pro Windows™ 95 a NT

## Popis jazyka pro scripty \*.TCP

Scripty jsou uloženy v běžných textových souborech, vytváříme je obvyklými textovými editory (např. programem Notepad.EXE). Zpracování souboru se děje striktně řádka po řádce, není prováděna žádná prekompilace ani není používáno bufferování či cache, takže je možné např. i průběžně modifikovat jeden script, který je opakovaně spouštěn z jiného. Struktura těch částí souboru, na něž se řízení toku příkazů nedostane, může být libovolná. Hlášení chyb syntaxe probíhá až v okamžiku, kdy řízení toku příkazů načte chybnou řádku.

Příkazem je vždy jedna celá řádka, není možné jeden příkaz rozdělit na více řádek nebo naopak uvést na jedné řádce více příkazů. Kdekoli mezi jednotlivými syntaktickými prvky je možné uvádět libovolné množství mezer a tabulátorů, je tedy možná i indentace bloků (mezerami před názvem příkazu).

Typickým parametrem příkazu je řetězec (viz Syntaxe řetězců), jméno souboru (viz Jméno souboru), adresa počítače, port. Konkrétní příkazy - viz Skupiny příkazů.

Prázdné řádky a řádky začínající znakem / (lomítko) jsou chápány jako komentáře. Jako komentář lze chápat i nedostupný úsek souboru (je-li před ním některý z příkazů změny řízení toku příkazů, tj. GoTo, Stop, End apod.).

# Vítejte do pomůcky k programu TCPBatch

Účelem programu je co nejvíce usnadnit vývojovým pracovníkům a systémovým operátorům simulovat subjekty TCP/IP spojení a tím odhalovat chyby a riziková místa programů komunikujících v sítích nebo hardwarové a instalační problémy těchto sítí.

Záměrem tvůrců nebylo umožnit přímo programování síťových aplikací, proto je zvolený scriptový jazyk velmi jednoduchý a zaměřený spíše na snadné a rychlé použití (již tří- až čtyřřádkové scripty mají své opodstatnění). Prvky tohoto jazyka byly vytvářeny s ohledem na konkrétní potřebu a zkušenosti obou jmenovaných skupin uživatelů tak, aby pokud možno jednotným způsobem a bez zdlouhavého pronikání do syntaxe a logiky jazyka byly pokryty všechny potřeby těchto uživatelů. Zvolený jazyk není neúměrně mocný, umožní však vše, co je pro tento typ případů potřeba.

Typickým uživatelem tohoto programu nebude programátor, který si podobnou simulaci může naprogramovat v mnohem mocnějším prostředí (Java, Perl, C++ apod.), kde může používat i komplikované datové struktury, větvení nebo rozsáhlou grafickou komunikaci s obsluhou.

Oproti uvedeným prostředím však program TCPBatch poskytuje tyto podstatné výhody:

- Nevyžaduje žádnou komplikovanou instalaci na počítači, kde bude použit. Lze si jej přinést na jedné disketě, z ní spustit a po ukončení práce s ním nezůstanou po něm záznamy v System Registry ani v adresáři počítače (doslova: nezbude po něm ani stopy).
- Jde o scriptové prostředí, takže lze napsat program během několika minut a bez složitého překládání ihned použít, navíc je možné pomocí externě definovaných symbolů značně parametrizovat již existující scripty.
- Umožňuje za běhu všestranně komunikovat s obsluhou.
- Nevyžaduje takřka žádnou průpravu, již krátké nahlédnutí do manuálu umožní uživateli tvořit použitelné programy, zvláště má-li již nějaké zkušenosti s programovacími jazyky či scriptovými interprety (Basic).
- Je cíleně vyvíjen pro výše zmíněnou skupinu uživatelů, proto není zatížen zbytečným balastem ve snaze o univerzálnost a příliš obecnými komplikovanými jazykovými prvky.
- Je jednoduchý, není třeba se jej učit (stačí si s sebou nosit tuto pomůcku TCPBATCH.HLP).
- Pracuje ve všech prostředích Windows™ 95 a Windows™ NT bez ohledu na použitou síťovou kartu, její ovladač nebo způsob připojení (pracuje úplně shodně i přes Slip/PPP připojení přes modem nebo sériový kabel).
- Je velmi malý a vejde se bez problémů na disketu s tools.

Při spuštění programu TCPBatch je v aktuálním adresáři vytvořen soubor pro protokol (TCPBATCH.LOG, pokud v příkazové řádce neuvedeme parametr -p), do něhož je opisována každá provedená akce programu a každý příkaz scriptu i s úplnou informací o jeho výsledku. Pokud dojde k nějaké chybě, je vypsána do souboru i s podrobným popisem situace, v níž nastala, a jejích pravděpodobných příčin. Soubor je v aktuální podobě čitelný MSDOS-ovým příkazem TYPE, programem Notepad.EXE nebo jinými prostředky, které soubor otevírají pouze pro čtení. Nenechte se vyvést z míry tím, že příkaz DIR nebo Explorer (Průzkumník) ukazují jeho nulovou délku.

Vyberte si oblast pomůcky:

- [Jak získat plnou verzi programu](#)
- [O aplikaci TCPBatch](#)
- [Jazyk scriptů \\*.TCP](#)
- [Skupiny příkazů](#)

Příkazová řádka programu  
Slovníček pojmů

## Způsoby zápisu jména souboru jako parametru příkazu

Pokud příkaz vyžaduje na určitém místě jako parametr jméno souboru, napíšeme zde souvislý řetězec nemezerových znaků bez jakýchkoli uvozovek nebo apostrofů a bez zvláštních znaků (např. znak \ nezdvojujeme).

Zápis jména souboru je zcela běžný, jaký známe z Windows<sup>TM</sup> nebo i MS-DOSu. Pokud je cesta k souboru v relativním zápisu (bez úvodního \ nebo prefixu "drive:", typicky prostě jméno souboru), odvozuje se skutečné jméno souboru od adresáře, v němž je uložen script, kde je tento příkaz uveden. V opačném případě je jméno na scriptu nezávislé.

Povolena jsou plná (dlouhá) jména souborů i jejich zkrácené 8+3 aliasy, znaky mimo obvyklou sadu (zvláště mezery) v názvech povoleny nejsou. Pokud soubor takové znaky obsahuje, musíme jej uzavřít do uvozovek (nejde však o řetězec, takže např. nezdvojujeme znaky backslash). Pokud soubor má obsahovat uvozovky, musíme je zdvojit.

Příklady zápisů názvu souborů (různých):

```
proc.tcp  
c:proc.tcp  
subdir\proc.tcp  
c:\rootdir\subdir\proc.tcp  
\\helga\rootdir\subdir\proc.tcp  
"c:\windows\media\Musica Exclamation.wav"  
"c:\tmp\Text ""1.A"""
```

TCP/IP spojení je plně duplexní a rovnocenné propojení dvou subjektů po síti internetovského typu. Uskutečňuje se tak, že jeden ze subjektů čeká na výzvu druhého, který se k němu připojuje na základě jednoznačného určení pomocí IP-adresy cílového počítače a čísla portu.

## Syntaxe řetězců - parametrů příkazů

Řetězcem rozumíme souvislý úsek znaků s binárními (ASCII) hodnotami 0 až 255. Je-li na místě parametru některého příkazu řetězec, čte se celý zbytek řádky a interpreтуje se jako spojení jednotlivých prvků řetězce (oddělených obvykle jednou nebo několika mezerami).

Prvkem řetězce může být:

**Znak nebo několik znaků v uvozovkách** vloží do řetězce přímo tyto znaky (bez oněch uvozovek). Výjimkou je znak backslash ('\, zpětné lomítko apod.), který není do řetězce přímo vkládán, ale ovlivňuje interpretaci dalšího znaku nebo několika znaků - následovat za ním může:

- **n** ... nový řádek (newline, LF, znak s kódem 10)
- **r** ... návrat vozíku (return, CR, znak s kódem 13)
- **f** ... nová stránka (formfeed, FF, znak s kódem 12)
- **e** ... escape (ESC, znak s kódem 27)
- **t** ... tabulátor (TAB, znak s kódem 9)
- **b** ... zvonek (bell, BEL, znak s kódem 7)
- číslo v desítkové soustavě (max. 3 číslice) ... znak s binární (ASCII) hodnotou

odpovídající tomuto číslu

- znak mimo výše uvedenou sadu ... tento znak samotný (bez ohledu na jeho speciální význam)

**Číslo v desítkové soustavě** ... do řetězce se vloží jeden znak s binární (ASCII) hodnotou odpovídající tomuto číslu

**Jedna z uvedených konstant** ... do řetězce se vloží jeden znak odpovídající převodní tabulce:

- |              |    |
|--------------|----|
| - <b>nl</b>  | 10 |
| - <b>lf</b>  | 10 |
| - <b>cr</b>  | 13 |
| - <b>esc</b> | 27 |
| - <b>tab</b> | 9  |

**Proměnná *line*** ... do řetězce se vloží naposledy načtená řádka. Proměnná je naplňována několika příkazy (způsob a přesný tvar naplnění proměnné je u každého takového příkazu uveden zvlášť) a má platnost až do dalšího takového příkazu. Při startu programu TCPBatch (nikoli scriptu) je proměnná nastavena na prázdný řetězec, modifikují ji příkazy Input, Receive, SkipTo a WaitFor.

**Jméno založeného čítače** ... do řetězce se vloží prostý dekadický zápis aktuální hodnoty tohoto čítače (např. "1", "25", "448"). Pokud čítač dosud nebyl založen (nebo také pokud text neodpovídá ani číslu, ani známému čítači, ani jedné z výše uvedených konstant a není ani v uvozovkách), nevloží se na tomto místě do řetězce nic, chyba se v takovém případě nehlásí. Viz také příkazy SetCounter a IncrementCounter.

### Příklady řetězců:

Následující řádky popisují stále tentýž řetězec:

"Ahoj\n"

```
"Ahoj" nl  
"Ahoj" 10  
"Ahoj\010"  
65 "hoj\n"
```

Po provedení příkazu:

```
SetCounter C_PocetRadek 110
```

budou následující řetězce shodné:

```
"Řádka č." "110" nl  
"Řádka č." C_PocetRadek nl  
"Řádka č.110\n"
```

Pokud byla během příkazu:

```
SkipTo nl
```

přijata od partnera řádka "220 Ready" ukončená znakem nl (což je v oblasti TCP/IP běžné ukončení řádky), budou následující řetězce shodné:

```
"Přijata řádka " line  
"Přijata řádka 220 Ready"
```

(vlastností příkazu SkipTo je, že do proměnné **line** neukládá ukončovací znak řádky).



## **Slovníček často používaných pojmů tohoto souboru**

TCP/IP spojení

Adresa počítače

Port

Definovatelný symbol

Návěští scriptu

Čítač

Check box

# Příkaz Connect

Syntaxe: **Connect** Adresa počítače Port

Program předpokládá, že na počítači identifikovaném touto adresou již čeká partnerský subjekt na spojení na tomto portu, a pokusí se k němu připojit. Pokud se to nezdaří do vypršení určitého časového limitu (závislého na systémovém nastavení), příkaz zahlásí chybu a ukončí všechny rozpracované scripty. V opačném případě je spojení navázáno a lze jej používat v následujících příkazech. Pokud bylo před provedením příkazu již nějaké spojení otevřeno, příkaz jej sám uzavře bez hlášení chyb.

Žádná z hodnot není v uvozovkách. Pokud je jedna z hodnot (nebo obě) otazník, otevře příkaz zvláštní dialogové okno a vyžádá si zadání nevyplněné hodnoty od obsluhy ve stejném formátu, v jakém jsou zadávány příkazu v příkazové řádce.

Příklady:

[Connect www.fbl.cz 80](#)

[Connect 195.25.85.11 25](#)

[Connect ? 4444](#)

Každý počítač v síti IP má svou jednoznačnou identifikaci, tzv. adresu, která je tvořena čtyřmi čísly 0-255 oddělenými tečkou. Kromě této adresy (nazýváme ji IP-adresou) může mít i několik DNS-záznamů (DNS-jmen), což je obvykle několik písmenno-číslicových shluků oddělených rovněž tečkami.

Portem rozumíme číslo 0-32767, které jednoznačně identifikuje proces čekající na spojení na cílovém počítači a obvykle i protokol (formát dat), kterým spolu budou subjekty TCP/IP spojení komunikovat. Standardní porty jsou např. 21 pro protokol ftp, 80 pro http, 25 pro smtp apod. Program TCPBatch nepodporuje symbolické názvy portů.

# Příkaz Accept

Syntaxe: **Accept Port**

Program přejde do stavu čekání na spojení na tomto portu, a čeká, až se nějaký partnerský subjekt (příkazem Connect nebo jiným kompatibilním způsobem) připojí. Program vždy čeká, dokud není spojení navázáno, pak jej lze používat v následujících příkazech. Po provedení příkazu Accept je port opět volný a může jej použít jiný program, a to bez ohledu na trvání takto navázaného spojení. Pokud bylo před provedením příkazu již nějaké spojení otevřeno, příkaz jej sám uzavře bez hlášení chyb.

Hodnota portu není v uvozovkách. Pokud je jí otazník, otevře příkaz zvláštní dialogové okno a vyžádá si její zadání od obsluhy ve stejném formátu, v jakém je zadávána příkazu v příkazové řádce.

Příklady:

[Accept 80](#)

[Accept 25](#)

[Accept ?](#)

Viz také: [Příkaz Connect](#)

# Příkaz Disconnect

Syntaxe: **Disconnect**

Program uzavře spojení otevřené dříve příkazem Connect nebo Accept. Pokud již spojení ukončil některý z předchozích příkazů nebo bylo ztraceno nebo ukončeno partnerským subjektem spojení, neprovede příkaz nic ani nehlásí žádnou chybu. Na konci scriptu ani před otevíráním dalšího spojení není nutné spojení uzavírat, program jej uzavírá sám (v případě ukončení všech rozpracovaných scriptů až po schválení obsluhou). Spojení je vždy ukončeno bez hlášení chyb.

Příklady:

Disconnect

Viz také: Příkaz Connect Příkaz Accept

# Skupiny příkazů v jazyce scriptů TCPBatch

V názvech příkazů můžeme libovolně kombinovat velká a malá písmena, tento rozdíl bude ignorován. Pokud patří jeden příkaz logicky do více skupin, je uveden ve všech takových skupinách.

## Příkazy pro manipulaci se spojením

Connect se připojí k čekajícímu subjektu spojení

Accept vytvoří čekající subjekt spojení

Disconnect uzavře běžící spojení

## Příkazy pro komunikaci s partnerským subjektem navázaného spojení

Send zašle partnerovi řetězec (blok dat)

Receive načte od partnera pevný počet znaků (blok dat)

SkipTo načte od partnera řetězec ukončený určitým znakem

WaitFor čte od partnera řádky, dokud nenarazí na koncovou podmínku

## Příkazy pro manipulaci se souborem pro opis přijímaných dat

OutputFile zahájí opis do souboru

CloseOutput ukončí opis do souboru

## Příkazy pro práci s množinou definovaných symbolů

Define zařadí symbol do množiny definovaných

Undefine vyřadí symbol z množiny definovaných

IfDef provede příkaz, pokud je symbol definován

IfNotDef provede příkaz, pokud není symbol definován

TestCounter porovná čítač s konstantou a na základě porovnání definuje symbol TRUE

TestCheck na základě stavu check boxu Označen/Neoznačen definuje symbol CHECKED

Question vyžádá od obsluhy odpověď Yes/No a podle ní definuje symbol CONSENT

## Příkazy pro změnu řízení toku příkazů scriptu

Label označí návěští ve scriptu

GoTo přesune tok příkazů na některé návěští

Call zavolá jiný (nebo i tentýž) script

Return ukončí tento script

Stop ukončí provádění všech rozpracovaných scriptů

End ukončí program TCPBatch

## Příkazy pro komunikaci s obsluhou

Signalize přehraje zvukový soubor \*.WAV

Message zobrazí zprávu a počká na stisk tlačítka OK

Question vyžádá od obsluhy odpověď Yes/No a podle ní definuje symbol CONSENT

Input načte řádku zadanou obsluhou pro další použití

WaitCheck počká na požadované označení check boxu obsluhou

## Příkazy pro práci s čítači

SetCounter založí čítač nebo mu změní hodnotu

IncrementCounter zvýší nebo sníží hodnotu čítače

TestCounter porovná čítač s konstantou a na základě porovnání definuje symbol TRUE

## Příkazy pro práci s check boxy

ShowCheckBox zobrazí nový check box

HideCheckBox zruší check box a uvolní místo pro jiný

SetCheck změní stav check boxu Označen/Neoznačen

TestCheck na základě stavu check boxu Označen/Neoznačen definuje symbol CHECKED

WaitCheck počká na požadované označení check boxu obsluhou

### **Ostatní příkazy**

Wait počká zadaný počet milisekund

ShowWindow zobrazí nebo skryje hlavní okno aplikace

Comment označuje komentář



## Příkaz Send

Syntaxe: **Send** Řetězec

Program odešle řetězec partnerskému subjektu spojení. Pokud již spojení ukončil některý z předchozích příkazů, bylo ztraceno či ukončeno partnerským subjektem nebo nebylo vůbec navázáno, příkaz zahlásí chybu a ukončí všechny rozpracované skripty. Příkaz zasílá řetězec v jeho čisté délce, nepřidává za něj konec řádky ani jej jinak neupravuje. Délka řetězce není omezena.

Příkaz v ideálním případě nečeká na fyzické odeslání řetězce, ale zařadí jej do odesílané fronty a program pak bezprostředně pokračuje dalším příkazem. Pouze pokud je odesílaná fronta již plná (při zasílání velkých objemů dat či při malé průchodnosti spojení), čeká příkaz nejprve na její uvolnění.

Příklady:

Send "220 Ready." nl

Send "254 Command " line " accepted." nl

# Příkaz Receive

Syntaxe: **Receive** počet znaků

Program přijme od partnerského subjektu spojení řetězec čítající právě tolik znaků. Pokud již spojení ukončil některý z předchozích příkazů, bylo ztraceno či ukončeno partnerským subjektem nebo nebylo vůbec navázáno, příkaz zahlásí chybu a ukončí všechny rozpracované skripty. Příkaz přijímá řetězec v jeho čisté délce (bez jakýchkoli konců řádky) a nemá vliv na event. následující čtení. Délka řetězce je omezena počtem 4096, je však možné číst i delší bloky prostým opakováním příkazu Receive.

Příkaz v ideálním případě nalezne znaky již přijaté ve vstupní frontě spojení, takže je pouze převezme a ihned se vrátí. Pokud partnerský subjekt dosud neodeslal dostatečný počet znaků, čeká příkaz nejprve na naplnění požadované délky vstupní fronty.

Pokud příkaz úspěšně načte řetězec, naplní jeho úplnou hodnotou systémovou proměnnou **line** (po příkazu 'Receive 5' bude mít **line** délku 5 znaků). Tato systémová proměnná pak může být použita jako součást řetězců (viz také).

V případě úspěšného načtení, je-li současně aktivní některý příkaz OutputFile, je řetězec v úplné délce zapsán do výstupního souboru (po příkazu 'Receive 5' se do souboru zapíše přijatých 5 znaků).

Příklady:

Receive 1

Receive 110

# Příkaz SkipTo

Syntaxe: **SkipTo** Řetězec

Řetězec je povinně jednoznakový (jinak příkaz zahlásí chybu a ukončí všechny rozpracované scripty). Program přijme od partnerského subjektu spojení řetězec znaků ukončených prvním (nejbližším) výskytem tohoto tzv. ukončovacího znaku. Pokud již spojení ukončil některý z předchozích příkazů, bylo ztraceno či ukončeno partnerským subjektem nebo nebylo vůbec navázáno, příkaz zahlásí chybu a ukončí všechny rozpracované scripty. Příkaz načte řetězec včetně zmíněného ukončovacího znaku (další event. příkazy pro čtení pokračují za tímto znakem). Délka řetězce je omezena počtem 4096 - pokud do tohoto počtu není přijat ukončovací znak, příkaz zahlásí chybu a ukončí všechny rozpracované scripty.

Příkaz v ideálním případě nalezne znaky již přijaté ve vstupní frontě spojení, takže je pouze převezme a ihned se vrátí. Pokud partnerský subjekt dosud neodeslal žádný ukončovací znak, čeká příkaz nejprve na jeho přijetí do vstupní fronty.

Pokud příkaz úspěšně načte řetězec, naplní jeho čistou hodnotou (bez ukončovacího znaku !) systémovou proměnnou **line** (po příkazu 'SkipTo "A"' a přijetí řetězce "HOLKA" bude mít **line** délku 4 znaky a hodnotu "HOLK"). Tato systémová proměnná pak může být použita jako součást řetězců (viz také).

V případě úspěšného načtení, je-li současně aktivní některý příkaz OutputFile, je řetězec v úplné délce (včetně ukončovacího znaku) zapsán do výstupního souboru (v našem příkladě se do souboru zapíše všech přijatých 5 znaků "HOLKA").

Typickým použitím příkazu je načtení jedné řádky příkazem **SkipTo nl** .

Příklady:

**SkipTo "A"**

**SkipTo nl**

**SkipTo "\n"**

# Příkaz WaitFor

Syntaxe:

**WaitFor End**  
**WaitFor EmptyLine**  
**WaitFor LineStarting** Řetězec  
**WaitFor ExactMatch** Řetězec

Program přijímá od partnerského subjektu spojení opakovaně řetězce znaků ukončené prvním (nejbližším) výskytem znaku **nl** (newline, ASCII hodnota 10). Pokud již spojení ukončil některý z předchozích příkazů, bylo ztraceno či ukončeno partnerským subjektem nebo nebylo vůbec navázáno, příkaz zahlásí chybu a ukončí všechny rozpracované scripty. K hlášení chyby a ukončení scriptů vede příkaz rovněž v případě, že ke ztrátě spojení či ukončení spojení partnerským subjektem dojde dříve, než některý z přijatých řetězců splní požadovanou podmínku (netýká se varianty **WaitFor End**). Příkaz načte každý řetězec včetně znaku **nl** (další event. příkazy pro čtení pokračují za tímto znakem). Délka každého řetězce je omezena počtem 4096 - pokud do tohoto počtu není přijat znak **nl**, příkaz zahlásí chybu a ukončí všechny rozpracované scripty.

Jednotlivé varianty se navzájem liší koncovou podmínkou, kterou musí naposledy načtený řetězec splňovat, aby se příkaz úspěšně ukončil a pokračovalo se dalším příkazem interpretovaného scriptu. Načítání skončí v případě, že naposledy načtený řetězec:

**End** - ... se nepodařilo úspěšně načíst, typicky partnerský subjekt ukončil spojení nebo bylo spojení přerušeno. Vedlejším efektem této varianty příkazu je uzavření spojení (není již třeba provádět Disconnect).

**EmptyLine** - ... je prázdný (s nulovou délkou). K řetězci se nepočítají znaky **cr**, které někdy mohou bezprostředně předcházet znak konce řádky **nl** (leckdy i ve více exemplářích).

**LineStarting** - ... začíná uvedeným řetězcem-parametrem.

**ExactMatch** - ... je přesně shodný s uvedeným řetězcem-parametrem. K řetězci se nepočítají znaky **cr**, které někdy mohou bezprostředně předcházet znak konce řádky **nl** (leckdy i ve více exemplářích).

V obou posledních příkazech musí být shoda včetně velkých a malých písmen a řetězec-argument musí být alespoň jednoznakový.

Pro každé jednotlivé čtení řetězce platí, že příkaz v ideálním případě nalezne znaky již přijaté ve vstupní frontě spojení, takže je pouze převezme a ihned se vrátí. Pokud partnerský subjekt dosud neodeslal žádný znak **nl**, čeká příkaz nejprve na jeho přijetí do vstupní fronty. Nikdy se netestují částečně přijaté řetězce.

Pokud je příkaz úspěšně proveden, naplní čistou hodnotou posledního řetězce (bez ukončovacího znaku !) systémovou proměnnou **line** (po příkazu 'WaitFor LineStarting "220"' a přijetí libovolného počtu řetězců nezačínajících znaky "220" a řetězce "220 Ready\n" bude mít **line** délku 9 znaků a hodnotu "220 Ready"). Tato systémová proměnná pak může být použita jako součást řetězců (viz také).

V případě úspěšného načtení každého přijatého řetězce, je-li současně aktivní některý příkaz OutputFile, je řetězec v úplné délce (včetně znaku **nl**) zapsán do výstupního souboru (v našem příkladě se do souboru zapíše všech přijatých 10 znaků "220 Ready\n").

Typickým použitím příkazu je načtení headeru e-mailového dopisu nebo dokumentu žádaného a posílaného protokolem http (ve tvaru **WaitFor EmptyLine**), čekání na odpověď ftp-serveru, kdy před řádkou začínající "220 " může být libovolný počet řádek začínajících "220-" apod. (ve tvaru **WaitFor**

`LineStarting "220 "`) nebo čekání na konec těla dopisu v protokolu smtp (ve tvaru `WaitFor ExactMatch ". "`).

Příklady:

- `WaitFor End`

- `WaitFor EmptyLine`

- `WaitFor LineStarting "Content-Type: "`

- `WaitFor LineStarting "250 "`

- `WaitFor ExactMatch ". "`

# Příkaz OutputFile

Syntaxe: **OutputFile** Jméno souboru

Příkaz zahájí binární opis přijímaných dat do souboru. Všechny znaky přijímané příkazy Receive, SkipTo a WaitFor budou zapisovány do tohoto souboru, přičemž nedochází k žádnému přidávání znaků (např. *cr* před každé *nl*) nebo naopak vynechávání znaků (např. konců řádek). Zápis do tohoto souboru končí příkazem CloseOutput, dalším OutputFile nebo ukončením všech právě rozpracovaných scriptů. Pokud by v tomto okamžiku zbývaly ještě nějaké znaky ve vstupní frontě spojení, již se do souboru nezapíší. Odesílaná ani obsluhou zadávaná data se do souboru v žádném případě neukládají.

Jméno souboru není v uvozovkách. Pokud se nezdaří soubor vytvořit a otevřít pro zápis, bude zhlášena chyba a ukončeno provádění rozpracovaných scriptů. Pokud již byl nějaký soubor dříve otevřen, bude před provedením příkazu nejprve uzavřen (není nutné explicitně uzavírat předchozí soubor pomocí CloseOutput).

Příklady:

[OutputFile c:\spool\tcpbatch.ptc](#)

[OutputFile \\irma\spool\tcpbatch.ptc](#)

[OutputFile LPT1:](#)

# Příkaz CloseOutput

Syntaxe: **CloseOutput**

Příkaz ukončí binární opis přijímaných dat do souboru. Pokud žádný soubor otevřen není, neprovede příkaz žádnou činnost. Nikdy nekončí chybou.

Viz také: [Příkaz OutputFile](#)

Příklady:

[CloseOutput](#)

Jméno symbolu tvoří souvislý úsek nemezerových znaků (velká a malá písmena nejsou ztotožňována). Symbol má globální platnost a může nabýt dvou stavů: Definován/Nedefinován. Na základě symbolů může docházet k větvení programu.



# Příkaz Define

Syntaxe: **Define** Definovatelný symbol

Příkaz zařadí symbol do množiny definovaných symbolů. Od tohoto okamžiku je symbol definován. Je lhostejné, zda byl symbol před provedením příkazu již definován či nikoli.

Význam definovanosti symbolů (tj. použití výsledku tohoto příkazu) je popsán v příkazech IfDef a IfNotDef.

Příklady:

Define ZaslatData

Define ModPrace\_1

Define \_signalizovat

# Příkaz Undefine

Syntaxe: **Undefine** Definovatelný symbol

Příkaz vyřadí symbol z množiny definovaných symbolů. Od tohoto okamžiku není symbol definován. Je lhostejné, zda byl symbol před provedením příkazu již definován či nikoli.

Význam definovanosti symbolů (tj. použití výsledku tohoto příkazu) je popsán v příkazech IfDef a IfNotDef.

Příklady:

```
Undefine ZaslatData  
Undefine ModPrace_1  
Undefine _signalizovat
```

# Příkaz Label

Syntaxe: **Label** **Návěští**

Příkaz neprovede žádnou činnost, pouze označuje místo v programu, kam lze provést přesun řízení toku příkazů příkazem GoTo. Pokud je ve scriptu více návěští téhož jména, má přednost to z nich, které je uvedeno na vyšší řádce (dříve).

Příklady:

Label Restart

Label Smyčka

Label KonecScriptu

Návěští je souvislý úsek nemezerových znaků, který identifikuje místo programu, kam lze provést skok příkazem GoTo. Velká a malá písmena nejsou ztotožňována. Je-li v jednom scriptu uvedeno více návěstí téhož jména, má přednost návěští uvedené dříve.

# Příkaz GoTo

Syntaxe: **GoTo** **Návěští**

Příkaz přeruší tok příkazů a vyhledá v programu návěští stejného jména (uvedené v příkazu Label), v případě jeho nalezení pokračuje interpretace scriptu za odpovídajícím příkazem Label. Pokud by bylo v jednom scriptu uvedeno více návěští téhož jména, dává se přednost tomu z nich, které je uvedeno dříve v souboru.

Pokud by v právě interpretovaném scriptu návěští stejného jména nebylo nalezeno, ukončí se provádění scriptu a návěští se hledá v tom souboru, z něžž byl tento script volán příkazem Call. Toto hledání se opakuje, dokud není návěští nalezeno nebo není uzavřen poslední rozpracovaný script - v tom případě se hlásí chyba a provádění je ukončeno. Tento způsob provádění příkazu GoTo je podobný technikám exceptionů či throw-catch a umožňuje snadné provádění restartů či výskoků z bloku i ve scriptech hluboko vnořených příkazy Call.

Příklady:

[GoTo Restart](#)

[GoTo Smyčka](#)

[GoTo KonecScriptu](#)

# Příkaz Call

Syntaxe: **Call** Jméno souboru

Příkaz přeruší provádění aktuálního scriptu a přesune tok příkazů na začátek scriptu s tímto jménem souboru. Pokud je ve vnořeném scriptu proveden příkaz Return nebo tok příkazů dorazí na konec souboru, vnořený script se uzavře a tok příkazů se přesouvá bezprostředně za příkaz Call.

Jméno souboru není v uvozovkách. Pokud se nezdaří soubor otevřít pro čtení (typicky protože vůbec neexistuje), bude zahlášena chyba a ukončeno provádění všech rozpracovaných scriptů.

Ne vždy se musí tok příkazů vrátit bezprostředně za příkaz Call - při způsobení zásadní chyby se ukončují všechny rozpracované scripty (tedy i tento), podobnou funkci mají příkazy Stop a End. Rovněž výskok příkazem GoTo může přesunout tok příkazů na jiné místo scriptu nebo i ven ze scriptu.

Doporučenou koncovkou (souborovou příponou) volaného scriptu je .TCP, tato koncovka se sama nedoplňuje. Volání scriptů navzájem je omezeno konstantou 80 a technickými možnostmi systému (neboť rozpracované scripty zůstávají jako soubory otevřené), pokud je tento počet překročen, bude zahlášena chyba a ukončeno provádění všech rozpracovaných scriptů. Při návratu (příkazem Return nebo jinak) je ukončený script jako soubor uzavřen a jeho pozice je uvolněna pro další script (počet příkazů Call v jednom scriptu není omezen).

Scripty se mohou volat i rekurzivně, neboť soubor je otevírán s možností čtení jinými procesy, je však nutno sladit hloubku rekurze s omezeními uvedenými v předchozím odstavci. Pokud je soubor prováděn, nelze jej editovat ani mazat, bezprostředně před příkazem Call jej však editovat lze, čehož je možné využít záměrně, pokud je potřeba v některém místě programu interaktivně měnit jeho část.

Příklady:

[Call malaakce.tcp](#)

[Call c:\tcpser\malaakce.tcp](#)

[Call \\irma\tcpser\malaakce.tcp](#)

# Příkaz Return

Syntaxe: **Return**

Příkaz ukončí provádění aktuálního scriptu, uzavře jej jako soubor a přesune tok příkazů bezprostředně za příkaz Call, který tento script volal. Pokud jde o script nejvyšší úrovně (tj. není volán z jiného scriptu, ale přímo z programu TCPBatch), je provádění ukončeno. Příkaz Return není nutno uvádět na úplném konci souboru.

Příklady:

**Return**

# Příkaz Stop

Syntaxe: **Stop**

Příkaz ukončí provádění všech rozpracovaných scriptů a vrátí se do hlavního okna programu TCPBatch.

Příklady:

**Stop**



# Příkaz End

Syntaxe: **End**

Příkaz ukončí provádění všech rozpracovaných scriptů i samotný program TCPBatch.

Příklady:

**End**

# Příkaz IfDef

Syntaxe: **IfDef** Definovatelný symbol Příkaz

Pokud je symbol definován (je v množině definovaných symbolů, kam byl zařazen příkazem Define nebo jiným způsobem), provede se příkaz uvedený za jménem symbolu. Na tento příkaz-parametr není kladeno žádné omezení, typicky jím bude GoTo, Call, Stop. Může jím ovšem být i další příkaz IfDef nebo IfNotDef, tímto způsobem lze provést nějaký příkaz při současném splnění několika podmínek na definovanost symbolů.

Pokud symbol definován není, ignoruje se celý zbytek řádky a příkaz neprovede žádnou akci.

Viz také: Příkaz IfNotDef

Příklady:

IfDef Nestartováno Define ZaslatData

IfDef KratkaData Call krdata.tcp

IfDef Skoncit Stop

IfDef Skoncit IfNotDef Pokracovat Stop

# Příkaz IfNotDef

Syntaxe: **IfNotDef** Definovatelný symbol Příkaz

Pokud není symbol definován (není v množině definovaných symbolů, kam nebyl vůbec zařazen nebo byl vyřazen příkazem Undefine nebo jiným způsobem), provede se příkaz uvedený za jménem symbolu. Na tento příkaz-parametr není kladeno žádné omezení, typicky jím bude GoTo, Call, Stop. Může jím ovšem být i další příkaz IfDef nebo IfNotDef, tímto způsobem lze provést nějaký příkaz při současném splnění několika podmínek na definovanost symbolů.

Pokud symbol definován je, ignoruje se celý zbytek řádky a příkaz neprovede žádnou akci.

Viz také: Příkaz IfDef

Příklady:

IfNotDef Startováno Define ZaslatData

IfNotDef DlouhaData Call krdata.tcp

IfNotDef Pokracovat Stop

IfNotDef Pokracovat IfDef Skoncit Stop

# Příkaz Wait

Syntaxe: **Wait** počet milisekund

Program počká odpovídající počet milisekund. Hodnota není shora omezena, musí jít o celé číslo. Ve skutečnosti není čekání zcela přesné, neboť se odvozuje od jednotek, kterými právě časuje operační systém. Čekání lze přerušit stiskem tlačítka "Přerušení scriptu".

Příklady:

Wait 1000

Wait 2500

# Příkaz Comment

Syntaxe: **Comment** jakýkoli text

Příkaz nevykoná žádnou činnost. Podobný význam mají řádky začínající znakem / (lomítko).

Příklady:

`Comment Zde začíná příjem holých dat`

`Comment Call recvdata.tcp`

Čítač je určen identifikátorem, souvislým shlukem nemezerových znaků. Velká a malá písmena nejsou ztotožňována. Čítač má celočíselnou hodnotu, může být zvětšován a zmenšován, lze mu nastavit novou hodnotu a jeho aktuální hodnotu v textové podobě lze vkládat do řetězců a porovnávat s číselnou konstantou. Čítače mají význam především, chceme-li odeslat či přijmout určitý počet řádek, oznamovat partnerovi spojení pořadí řádky apod.

# Příkaz SetCounter

Syntaxe: **SetCounter** Čítač Hodnota

Uloží do proměnné-čítače novou hodnotu. Pokud čítač dosud nebyl definován, založí jej. Nikdy nezpůsobí chybu.

Žádný z parametrů není v uvozovkách.

Příklady:

SetCounter C\_PočetŘádek 1

SetCounter C\_Zbývá 110

# Příkaz IncrementCounter

Syntaxe:

**IncrementCounter** Čítač

**IncrementCounter** Čítač Hodnota

Zvýší proměnnou-čítač o uvedenou hodnotu. Pokud je hodnota záporná, bude hodnota čítače ve skutečnosti snížena. Neuvedeme-li hodnotu, zvýší se čítač o 1. Pokud čítač dosud nebyl definován, způsobí chybu a ukončí všechny rozpracované skripty.

Žádný z parametrů není v uvozovkách.

Příklady:

IncrementCounter C\_PočetŘádek

IncrementCounter C\_Celkem 5

IncrementCounter C\_Zbývá -1



# Příkaz TestCounter

Syntaxe:

**TestCounter** Čítač [ < | > | <= | >= | = | <> ] Hodnota

Porovná proměnnou-čítač s uvedenou hodnotu běžným způsobem (<> znamená neshodu). Pokud je podmínka splněna, zařadí symbol **TRUE** do množiny definovaných symbolů (viz příkaz Define). Pokud podmínka splněna není, vyřadí symbol **TRUE** z množiny definovaných symbolů (viz příkaz Undefine). Pokud čítač dosud nebyl definován, způsobí chybu a ukončí všechny rozpracované skripty.

Žádný z parametrů není v uvozovkách. Parametry musí být tři a mezi každými dvěma musí být aspoň jedna mezera.

Příklady:

TestCounter C\_PočetŘádek > 5

TestCounter C\_Celkem <= 75

TestCounter C\_Zbývá <> 0

Check box je prvek ovládacího panelu programu s hodnotami Označen/Neoznačen. Během práce scriptu může obsluha měnit jeho hodnotu, změnu označení lze provádět i příkazy scriptu, kterými se rovněž check boxy zobrazují a skrývají. Check box je určen libovolným shlukem nemezerových znaků, malá a velká písmena se neztotožňují. Současně může být zobrazeno max. 10 různých check boxů.

# Příkaz ShowCheckBox

Syntaxe: **ShowCheckBox** Check box Řetězec

Pokud check box s tímto jménem dosud nebyl zobrazen, zobrazí jej na prvním volném místě s hodnotou Neoznačen a popisným textem daným řetězcem-parametrem. Pokud byla překročena maximální hodnota počtu současně zobrazených check boxů (10), zahlásí příkaz chybu a ukončí provádění všech rozpracovaných scriptů.

Pokud byl již check box s tímto jménem zobrazen, neprovede příkaz žádnou akci, především nemění označení ani popisný text.

Je-li řetězec-parametr prázdný (nebo není vůbec uveden), bude popisný text shodný se jménem check boxu.

Příklady:

ShowCheckBox CB\_Větvit "Má se větvit"

ShowCheckBox CB\_OK "Hotovo"

ShowCheckBox CB\_OK

# Příkaz HideCheckBox

Syntaxe: **HideCheckBox** Check box

Pokud byl check box s tímto jménem zobrazen, skryje jej a uvolní jeho místo pro jiný check box. V opačném případě neprovede příkaz žádnou akci.

Příklady:

HideCheckBox CB\_Větvit

HideCheckBox CB\_OK

# Příkaz SetCheck

Syntaxe: **SetCheck** **Check box** [ **0** | **1** | **2** ]

Příkaz změní stav, hodnotu check boxu s tímto jménem (tzn. je/není označen). Na základě druhého parametru bude check box po provedení příkazu:

**0** - neoznačen

**1** - označen

**2** - změněno označení (z označen na neoznačen a naopak)

Pokud check box s tímto jménem dosud nebyl zobrazen, zahlásí příkaz chybu a ukončí provádění všech rozpracovaných scriptů.

Označení check boxu může ovšem libovolně měnit i obsluha (myši, klávesnicí).

Příklady:

SetCheck CB\_Větvit 0

SetCheck CB\_OK 1

SetCheck CB\_OK 2

# Příkaz TestCheck

Syntaxe:

**TestCheck** Check box

Pokud je check box tohoto jména označen, zařadí symbol **CHECKED** do množiny definovaných symbolů (viz příkaz Define). Pokud check box není označen nebo vůbec nebyl zobrazen příkazem ShowCheckBox, vyřadí symbol **CHECKED** z množiny definovaných symbolů (viz příkaz Undefine).

Příkaz nikdy nezpůsobí chybu.

Příklady:

TestCheck CB\_OK

TestCheck CB\_Pokračovat

# Příkaz WaitCheck

Syntaxe:

**WaitCheck Check box**

**WaitCheck Check box [ 0 | 1 ]**

Příkaz čeká, dokud obsluha nezruší označení (v případě druhého parametru **0**) resp. neoznačí (v případě druhého parametru **1** nebo při jeho vynechání) check box s tímto jménem. Pokud již před provedením příkazu měl check box požadovanou hodnotu, program ihned pokračuje dalším příkazem.

Čekání lze přerušit i stiskem tlačítka "Přerušení scriptu", které ukončí všechny rozpracované scripty.

Pokud check box s tímto jménem dosud nebyl zobrazen, zahlásí příkaz chybu a ukončí provádění všech rozpracovaných scriptů.

Příklady:

WaitCheck CB\_Větvit 0

WaitCheck CB\_OK 1

WaitCheck CB\_OK

# Příkaz Signalize

Syntaxe: **Signalize** Jméno souboru

Příkaz přehraje WAVE-soubor s uvedeným jménem na zvukovém zařízení. Způsob závisí na nainstalování zvukových ovladačů v systému. Není-li možné soubor přehrát, neproveden příkaz nic. Nečeká se na dokončení přehrání a ihned se pokračuje dalším příkazem.

Jméno souboru není v uvozovkách. Příkaz nikdy nehlásí chybu.

Příklady:

[Signalize c:\media\prihlas.wav](#)

[Signalize \\irma\media\prihlas.wav](#)



# Příkaz Message

Syntaxe: **Message** Řetězec

Program zobrazí řetězec v dialogovém okně s tlačítky OK a Cancel a přeruší zpracovávání scriptu, dokud obsluha nestiskne jedno z nich. Pokud obsluha stiskne OK, pokračuje script dalšími příkazy, pokud stiskne Cancel, ukončí se všechny rozpracované scripty, jakoby bylo stisknuto tlačítko "Přerušení scriptu" v hlavním panelu programu TCPBatch. Délka řetězce-parametru není shora omezena, řetězec musí mít aspoň jeden znak.

Pokud je hlavní okno aplikace skryto nebo zakryto jinými okny, prosadí se dialogové okno do popředí všech oken, aby nemohlo být přehlédnuto.

Příklady:

Message "Program dospěl až do bodu 2"

Message "Přišla nám tato řádka: " line

Message "Hodnota čítače je nyní: " C\_PocetRadek

# Příkaz Input

Syntaxe: **Input** Řetězec

Program zobrazí řetězec v dialogovém okně s prostorem pro zadání jedné řádky textu a tlačítka OK a Cancel a přeruší zpracovávání scriptu, dokud obsluha nezadá nějaký text (může být i prázdný) a nestiskne jedno z tlačítek. Pokud obsluha stiskne OK, pokračuje script dalšími příkazy, pokud stiskne Cancel, ukončí se všechny rozpracované scripty, jakoby bylo stisknuto tlačítko "Přerušení scriptu" v hlavním panelu programu TCPBatch. Délka řetězce-parametru není shora omezena, řetězec musí mít aspoň jeden znak.

Načtený řetězec je uložen do systémové proměnné *line*. Tato systémová proměnná pak může být použita jako součást řetězců (viz také).

Pokud je hlavní okno aplikace skryto nebo zakryto jinými okny, prosadí se dialogové okno do popředí všech oken, aby nemohlo být přehlédnuto.

Příklady:

Input "Zadejte řádku pro odeslání partnerovi"

Input "Zadejte odpověď na příšlou řádku: " line

# Příkaz ShowWindow

Syntaxe: **ShowWindow** [ 0 | 1 ]

Příkaz nastaví viditelnost hlavního okna aplikace na pracovní ploše podle druhého parametru takto:

0 - okno bude skryto (minimalizováno), nebude vidět

1 - okno bude viditelné v normální velikosti

Účelem příkazu je uklidit pracovní plochu v době, kdy script nevykonává žádnou činnost, při níž by měl být kontrolován obsluhou. Pokud všechny rozpracované scripty jakýmkoli způsobem skončí a program TCPBatch přejde do stavu výběru prováděného scriptu, okno se vždy zobrazí v normální velikosti. Pokud script narazí na některý z příkazů vyžadujících interakci s obsluhou (Message, Input apod.), je dotaz zobrazen v popředí pracovní plochy, ale po odpovědi obsluhy zůstává hlavní okno aplikace v původní viditelnosti (bylo-li skryto, zůstane skryto).

Příkaz nemůže způsobit žádnou chybu. Pokud již bylo okno předtím viditelné požadovaným způsobem, neprovede příkaz žádnou akci.

Viditelnost hlavního okna aplikace může ovšem libovolně měnit i obsluha (myší, klávesnicí).

Příklady:

ShowWindow 0

ShowWindow 1

# Příkaz Question

Syntaxe: **Question** Řetězec

Program zobrazí řetězec v dialogovém okně s tlačítky Yes, No a Cancel a přeruší zpracovávání scriptu, dokud obsluha nestiskne jedno z nich. Pokud obsluha stiskne Yes nebo No, pokračuje script dalšími příkazy, pokud stiskne Cancel, ukončí se všechny rozpracované scripty, jakoby bylo stisknuto tlačítko "Přerušení scriptu" v hlavním panelu programu TCPBatch. Délka řetězce-parametru není shora omezena, řetězec musí mít aspoň jeden znak.

Pokud je stisknuto tlačítko Yes, zařadí symbol **CONSENT** do množiny definovaných symbolů (viz příkaz Define). Pokud je naopak stisknuto No, vyřadí symbol **CONSENT** z množiny definovaných symbolů (viz příkaz Undefine).

Pokud je hlavní okno aplikace skryto nebo zakryto jinými okny, prosadí se dialogové okno do popředí všech oken, aby nemohlo být přehlédnuto.

Příklady:

Question "Mám spustit podřízený script ?"

Question "Přišla nám tato řádka: " line " - opakovat čtení ?"

Question "Hodnota čítače je nyní: " C\_PocetRadek " - pokračovat ?"

# Parametry příkazové řádky programu

Normálně spouštíme program bez parametrů. Za určitých okolností (zejména pokud nespouštíme jednoúčelový právě napsaný script, ale nějaký komplexnější) však může být výhodné nezahájit zpracování scriptu "s čistým štítem", ale mít již připraveny některé hodnoty, definice, nebo dokonce mít již navázáno spojení.

Každý z parametrů má předepsaný povinný počet hodnot. Pro parametry platí, že mezi hodnotami i za poslední z nich musí být aspoň jedna mezera, první z hodnot od písmena určujícího typ parametru může a nemusí být oddělena mezerou. Lze tedy psát rovnocenně **-d NEZAHAJIT** i **-dNEZAHAJIT**. Před písmenem určujícím typ parametru musí být bezprostředně pomlčka, jinak je tento parametr již sám o sobě považován za jméno autostartového scriptu a všechny následující parametry jsou ignorovány.

Pokud je za poslední hodnotou posledního parametru (je-li vůbec nějaký uveden) ještě nějaký text, je považován za jméno souboru se scriptem \*.TCP a program TCPBatch se jej snaží ihned (bez čekání na stisk nějakého tlačítka) spustit. Pokud se soubor nezdaří najít nebo otevřít, je zhlášena chyba a parametr je ignorován. Tomuto souboru říkáme autostartový script. Používá se obvykle, pokud má být nějaká akce spouštěna často ve stejném kontextu (třeba už při startu počítače). Obvykle je autostartový script ukončován příkazem End a obsluha se o něj vlastně vůbec nestará. Při shutdownu (ukončení systému před vypnutím počítače nebo při restartech) není nutno explicitně ukončovat program TCPBatch, ten odumře sám bez jakýchkoli škod.

Parametry mohou být:

**-p** Jméno souboru změní jméno souboru-protokolu, kam jsou vypisovány jednořádkové záznamy o všech prováděných akcích a příkazech od spuštění programu TCPBatch. Neuvedeme-li parametr -p, bude tímto souborem TCPBATCH.LOG v aktuálním adresáři. Je-li místo jména souboru pomlčka (tj. uvedeme ve tvaru -p-), nikam se nic neprotokoluje.

**-d** Definovatelný symbol definuje symbol pro možná větvení scriptů příkazy IfDef/IfNotDef podobným způsobem, jako to dělá příkaz Define.

**-n** Čítač **Hodnota** vytvoří nový čítač a nastaví mu počáteční hodnotu (celé číslo) podobným způsobem, jako to dělá příkaz SetCounter. Oba parametry musí být od sebe odděleny mezerou.

**-l** **Hodnota** uloží hodnotu (shluk nemezerových znaků) jako počáteční hodnotu systémové proměnné line, kterou lze dále používat jako součást řetězců. Normálně je tato proměnná při startu programu prázdná.

**-c** Adresa počítače Port provede před startem programu obdobu příkazu Connect s týmiž parametry, takže script je zahájen už s navázaným spojením. Otazníky zde nejsou povoleny.

**-a** Port provede před startem programu obdobu příkazu Accept s týmiž parametrem, takže script je zahájen už s navázaným spojením. Otazník zde není povolen.

Příklady kombinací parametrů:

```
-p- -c cvut.cz 25 -l Eleps  
-p c:\spool\http.log -a 80 -n C_PocetRadek 0  
-p- -dNavazSpojeni -dStaleOpakovat c:\tcpscr\spoj80.tcp  
\\helga\tcpscr\spoj80.tcp
```

## Možnosti získání plné verze programu TCPBatch

Tato verze programu je shareware. Oproti plné verzi programu se liší pouze tím, že během jednoho spuštění scriptu máte k dispozici pouze libovolných (Vámi zvolených) šest z celé sady příkazů jazyka.

Shareware-verzi lze libovolně používat i distribuovat bez porušení autorského zákona, tzn. pokud nebude jakkoli upravována (měněna) a pokud bude šířena spolu se všemi doprovodnými soubory, zejména s tímto souborem pomůcek. Časové omezení použití není zvlášť upraveno.

Plnou verzi programu lze zakoupit u výhradního distributora, FBL Group, spol. s r.o. na adrese  
FBL Group  
Petrohradská 18  
101 00 Praha 10

Informace o licenčních podmínkách a ceníky získáte na internetové adrese (URL) ***<http://www.fbl.cz>*** nebo na pražském telefonním čísle 7174 5165, kontakt je možný rovněž písemně na výše uvedené adrese a elektronickou poštou na adrese ***[tcpbatch@fbl.cz](mailto:tcpbatch@fbl.cz)***.



