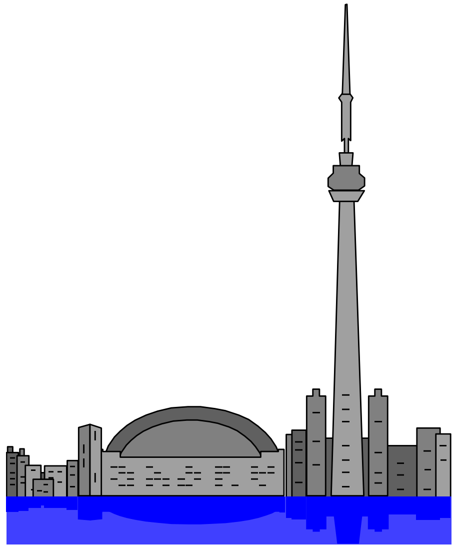
Programming the Command Interpreter

A Robelle Presentation

Interex 1995

Toronto, Canada

August 15 - 18, 1995

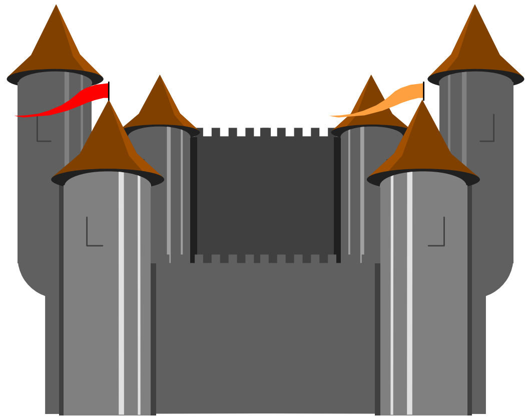


Topics covered

- A little history
- Variables
- Program flow
- Command files
- Tricks
- Where you can start

A little history

- HP 2000
    - O/S was BASIC interpreter
    - All work done with programs
- HP 3000 MPE/V
    - Job control shell
    - Only variables were JCWs
- HP 3000 MPE/iX
    - Advanced shell
    - Variables include integers, strings and boolean
    - Command files are programs



Variables

- Three kinds of variables:
  - Integer

    +/- 2^{31} (-2,147,483,648 to 2,147,483,648)
  - Boolean

    True or False
  - String

    Up to 255 characters

Naming rules for variables

- Max length is more characters than you want to retype
- Can have alphanumeric and underbar characters
- Must start with alpha or underbar
- Not case sensitive

```
total_count
love_potion_no_9
In_The_Big_Inning
```

Referencing variables

- Prefix the name with "**!**"

```
      !when_hell_freezes_over
```

- Multiple exclamation marks !!!!
  - Result depends on number of "**!**" left after dividing by 2
  - If number of "**!**" left is odd, value of variable is returned

```
    !!!!!burp                    !value_of_burp
```

returns

  - If even, variable name is returned

```
    !!!!burp                     !!burp
```

returns

Manipulating variables

- You can use these commands:

```
setvar expression
input var_name ;prompt=prompt ;wait=secs
deletevar var_name
showvar var_name
```

- HP pattern match available for DELETEVAR and SHOWVAR

```
showvar hp@
deletevar te#p?x@
```

Types of expressions

- Strings

```
"Flamingo Row" + "!color" - "PINK"
```

- Integer

```
3700 - (!total_amount / !total_count)
```

- Boolean

```
!batch_job and !hpjobcount < 10
```

Functions

- You can use functions in expressions

```
if lft("!in_str"),1) = "Y" then
setvar pos_val abs(!in_val - 10)
if typeof(my_var) = 0 then
```

- See Appendix C in volume 2 of the *Command Interpreter Manual*

- Many powerful functions

```
ups (string)
str (string, start, count)
pos (find_string, source)
finfo ("file_name", function)
```

upshift string

extract string

locate string

varies according to function

Flow control

- You can use these commands:

  if *expression* then

  else

  elseif

  endif

  while *expression*

  endwhile

```
    if "!hpjobname" = "MANAGER" then
    setvar line_count 0
    while !line_count < 10
        setvar line_count !line_count + 1
        echo Line count is !line_count
    endwhile
  endif
```

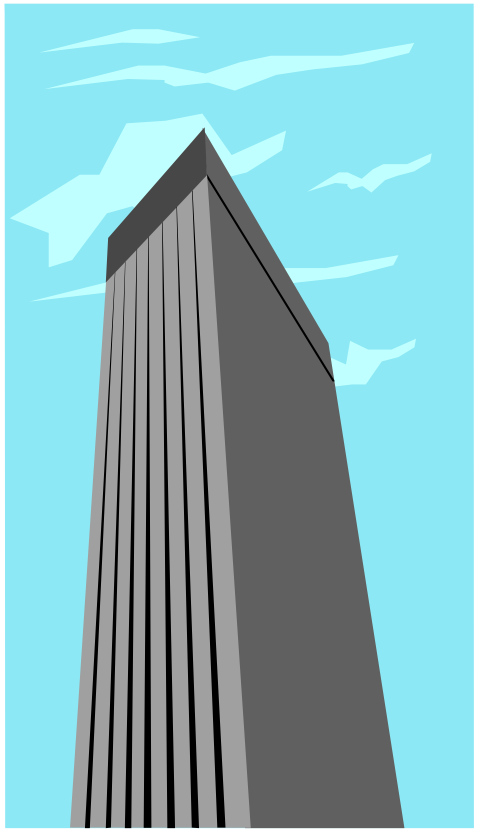
Command files

- Like good ol' UDCs
- Collection of many commands in one file
- Command files are useful
  - Hide details from user
  - Reduce repetitive tasks
- File attributes
  - Can be variable or fixed length
  - Need R and X access
  - Can be MPE or POSIX file space

Command file structure

```
parm local parameter declaration
   anyparm
   option declaration
   MPE command
   .
   .
   MPE command
```

- Parm lines are optional
    - ANYPARM must be last
- Options
    - Nobreak
    - Nolist



Example of command file

- Getit.cmd

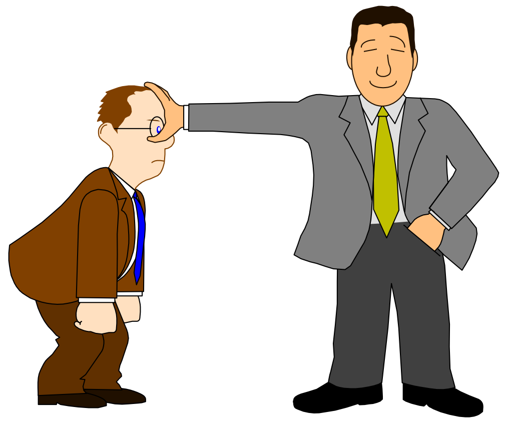
```
parm amt=0,name="Madame Zelda"
if !amt=0 then
     echo usage:  getit amt,name
     return
endif
setvar total_amt = !amt * 20
echo The total would be !total_amount, !name
```

- Note the RETURN command exits the current command file

More flow control

Two more commands that work in command files:

- RETURN command exits the current command file
- ESCAPE *number*
    - Exits nested command files back to the CI
    - Sets CIERROR to specified number (optional)
    - Also works interactively



More about command files

- Two ways to invoke command file
    - Type the name (follows Hppath)
    - Call from another command file
- Parms can be specified by name as well as position
    - Parm line in Setit.cmd

```
parm count,desc,flavor
```

    - For example,

```
setit flavor="tiger tiger"
```

Local variables

- Create with the parm line
- Can be string, numeric or boolean
- Can only be referenced in an expression
- Valid as long as the command file is executing

CI Programming Tip: Don't create local and global variables with the same name.

```
parm hpjobname="ZZTOP"
showvar hpjobname
```

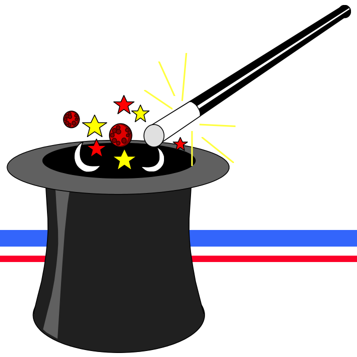
I/O redirection

- Works with MPE/iX 4.0 or higher
- Use ">" to redirect output
- Use "<" to redirect input
- Use ">>" to redirect output and append to file

```
echo The time is !hptime > tempfile
echo The date is !hpdate >> tempfile
```

- Output goes to temporary file
    - You can override this default with a file equation

Trick #1



- Setting a variable to data from a file
    - Use the INPUT command
    - Copies the first record from the file

```
input my_var < tempfile
setvar my_var rtrim(my_var)
```

- What about data in other records?
    - Create a second file from the first

```
print tempfile;start=2;end=2 > tempfil2
input my_var_2 < tempfil2
```

Variable dereferencing

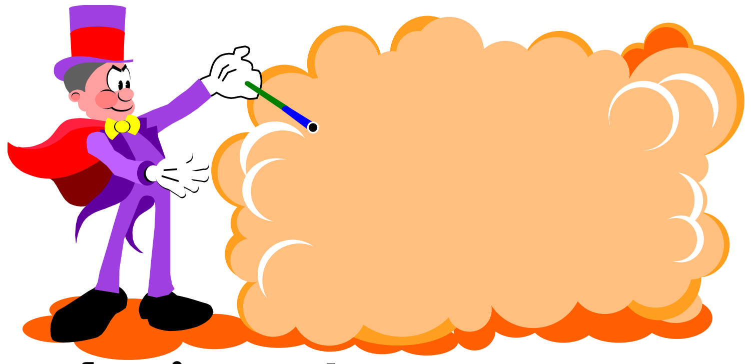
- Create dynamic output by careful dereferencing

```
setvar hpprompt "!!hpgroup !!hptime:"
```

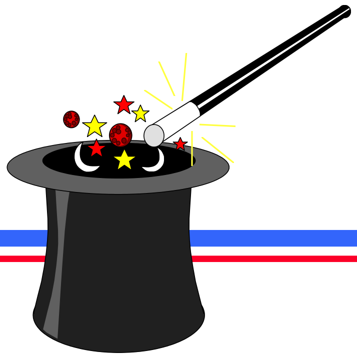
This sets hpprompt to "!hpgroup !hptime:"

- Use compound variables to simulate arrays

```
setvar  cnt 5
setvar  month_!cnt "May"
echo    Month !cnt contains !'month_!cnt'
```



Trick #2



- Dynamic code
    - Variables can be commands
    - Tricky dereferencing

```
          setvar my_cmd "setvar apple ""orange"""
          !my_cmd
```

- Also works with the SETVAR and INPUT commands

```
          setvar month_no 0
     while !month_no < 12
          setvar month_no !month_no + 1
          input month!month_no,                 &
               "Enter nbr days in month !month_no>"
     endwhile
```

Reincarnation of variables

- Variable lifetime only as long as session
- Save variables to file and execute file at logon time

```
        purge varsave
        setvar month_no 0
        while !month_no < 12
              setvar month_no !month_no + 1
              echo setvar month_!month_no !month_!month_no
   &
                >> varsave
        endwhile
        save varsave
```

- Invoke the file from your logon UDC

grep

- Fast utility ported from UNIX to find strings in files
- Only available on MPE/iX 5.0 or higher
- For more documentation, use **`man grep`** within shl.bin.sys
- Phone.cmd

```
        anyparm in_name
        run grep.hpbin.sys;                 &
                                            info="-i '!
          in_name', PHONES"
```

- Filename must be in uppercase for MPE name space
- Search file must be permanent

Using command files inside Qedit

- Most MPE commands available from Qedit
  - Not BYE or HELLO
- Execution of command files follows Hppath
- 10 levels of recursion instead of 30
- Some commands are simulated
  - Exit to CI with CI command
- Use Qedit commands inside MPE command files
  - Prefix Qedit commands with a "/"

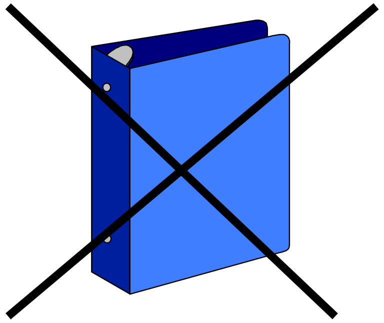
```
/listqj !line_no
/text !in_file
```

Undocumented Qedit Trick (:/)

- To redirect I/O, prefix Qedit commands with a ":"
- Must have **`/set list init off`**
- Works on *any* Qedit command

```
:/list !line_no > tempfile
:/del first/* > $null
:/ver zz > tempzz
```

  - If zz = 1/10, Tempzz will contain **`SET ZZ 1/10`**
  - To restore setting, just invoke Tempzz



How to learn programming the CI

```
while !awake
     !program_the_ci
endwhile
```