IBC233 - System i Business Computing

Week 8: Creating CLLE command & RPGLE Programming with database files

Agenda

- Creating a CLLE command
- RPGLE Programming with database files

CL Command Review

PGM	starts a CL program and defines parameter list
DCL	Declares variables: data types can be Decimal (*DEC), Character (*CHAR), Logical (*LGL) Signed Integer(*INT) and Unsigned Integer(*UINT)
DCLF	Declare a file - physical, logical or display attribute
RCVF	receives a record from a display, physical or logical file
SNDRCVF	- sends a record format to user's terminal with current states of indicators and output fields, - waits for the user to input data and press Enter or a function key, - receives input fields and response indicators into the program.
RTVOBJD	- Retrieve Object Description. Can find object owner, or the level of the system when the object was created.
CHGVAR	Change the contents of a variable
GOTO	Goes to another part of the program
MONMSG	Monitor Message - checks for operating system messages
DOWHILE	Executes a loop until a condition is met. The condition is tested at the beginning of the loop.
DOUNTIL	Executes a loop until a condition is met. The condition is tested at the end of the loop.
SELECT- ENDSELECT:	Tests multiple conditions
IF-ELSE	Tests one condition.
DO	Start of a logical block.
ENDDO	Ends a logical block or a loop.
ENDPGM	ends a CL program

Built-in Function: %SST

- A substring function in CL
- %SST(&variable start length)

```
Dcl &var1  *char  len(20) value('IBC233')
Dcl &var2  *char  len(5)

Chgvar &var1  (%sst(&var1 4 3))
```

What is the value of &var1?

Built-in Function: %trimr

- Used to remove trailing spaces from a character string in RPG!

```
fname = 'Cindy    ';
lname = 'Laurin   ';
```

Then

```
fullname = %trimr(fname) + lname;
```

what would be the value of fullname?

- More built-in function on IBM website

CL Command: DSPOBJD - RTVOBJD

- Shows/retrieves compile information about an object.
  - Date compiled
  - Version of the operating system used when the object was compiled.
- QCMD *PGM object
  - A special object – it's OS version of the object can reflect current operating system level.

CL Command: DSPCMD

- Displays information about a command including where the source code is!

Passing Parameters

- CL program (e.g. PARMCL) code:

```
pgm (&parm1  &parm2)

 dcl &parm1 *dec (5 2)
 dcl &parm2 *char 5
```

- To call this program

```
CALL PARMCL (X'10002F'  'TEST')
```

CL Commands

- A CL command consists of a program with a compiled object.
- Command Source
  - Builds the command entry screen
  - Calls the compiled object
  - Interpreted

Creating CL Commands

- To create/define a user-defined command, you need to:
  - enter command definition statements into a source file and compile them into an commend object,
  - run a Create Command (CRTCMD) command using the object and an program as inputs.

Command Definition

- The command definition of each command contains one or more **command definition statements**:
  - One and only one Command (CMD) statement
  - A Parameter (PARM) statement for each parameter that appears on the command being created.
  - Other optional statements, such as DEP, ELEM, QAUL statements.

Syntax for Command Definition

```
CMD   PROMPT(  )

PARM  KWD ( )   +
      MIN( )    +
      TYPE( )   +
      LEN( )    +
      RSTD( )   +
      VALUES( ) +
      DFT( )    +
      PROMPT( )
```

PARM Reference:

Keyword	Description
KWD	Keyword
TYPE	Type of value
MIN	Minimum values required
LEN	Value length
RSTD	Restricted values
DFT	Default value
VALUES	Valid values
PROMPT	Prompt specifications

Example: HELLOCMD.cmd

```
CMD 'HELLO UNIVERSE'

PARM    KWD(PLANET)                                    +
        MIN(1)                                         +
        TYPE(*CHAR) LEN(8)                             +
        PROMPT('DESTINATION PLANET:')
PARM    KWD(MYNAME)                                    +
        MIN(1)                                         +
        TYPE(*CHAR)  LEN(31)                           +
        PROMPT('MY NAME:')
PARM    KWD(OUTPUT)                                    +
        TYPE(*CHAR) LEN(9)                             +
        RSTD(*YES)                                     +
        VALUES(*MSGLINE *DISPLAY *PRINTER)             +
        DFT(*MSGLINE)                                  +
        PROMPT('OUTPUT FOR UNIVERSE MESSAGE:')
PARM    KWD(SHOWNME)                                   +
        TYPE(*CHAR)  LEN(4)                            +
        RSTD(*YES)                                     +
        VALUES(*YES *NO)                               +
        DFT(*NO)                                       +
        PROMPT('SHOW NAME:')
```

Example: HELLOCPP.clle

```
PGM         (&PLANET  &FULLNAME &OUTPUT &SHOWNAME)

DCLF        HELLODSP

DCL         &SHOWNAME     *CHAR    4
DCL         &OUTPUT       *CHAR    8

SELECT
   WHEN        (&OUTPUT = '*MSGLINE')  DO
      IF          (&SHOWNAME = '*YES') DO
         SNDPGMMSG  MSG('A Hello Universe to'  *BCAT &FULLNAME          +
                        *BCAT 'traveling to'           *BCAT &PLANET )   +
                        MSGTYPE(*COMP)
      ENDDO
      ELSE        DO
         SNDPGMMSG  MSG('Hello universe to the person'  *BCAT 'traveling to'  +
                        *BCAT &PLANET )   MSGTYPE(*COMP)
      ENDDO
   ENDDO
   WHEN        (&OUTPUT = '*DISPLAY')  DO
      IF          (&SHOWNAME = '*YES') CHGVAR &IN99 '1'
      ELSE        CHGVAR &IN99 '0'

      SNDRCVF     RCDFMT(RECORD1)
   ENDDO
ENDSELECT
ENDPGM
```

Example: HELLODSP.dspf

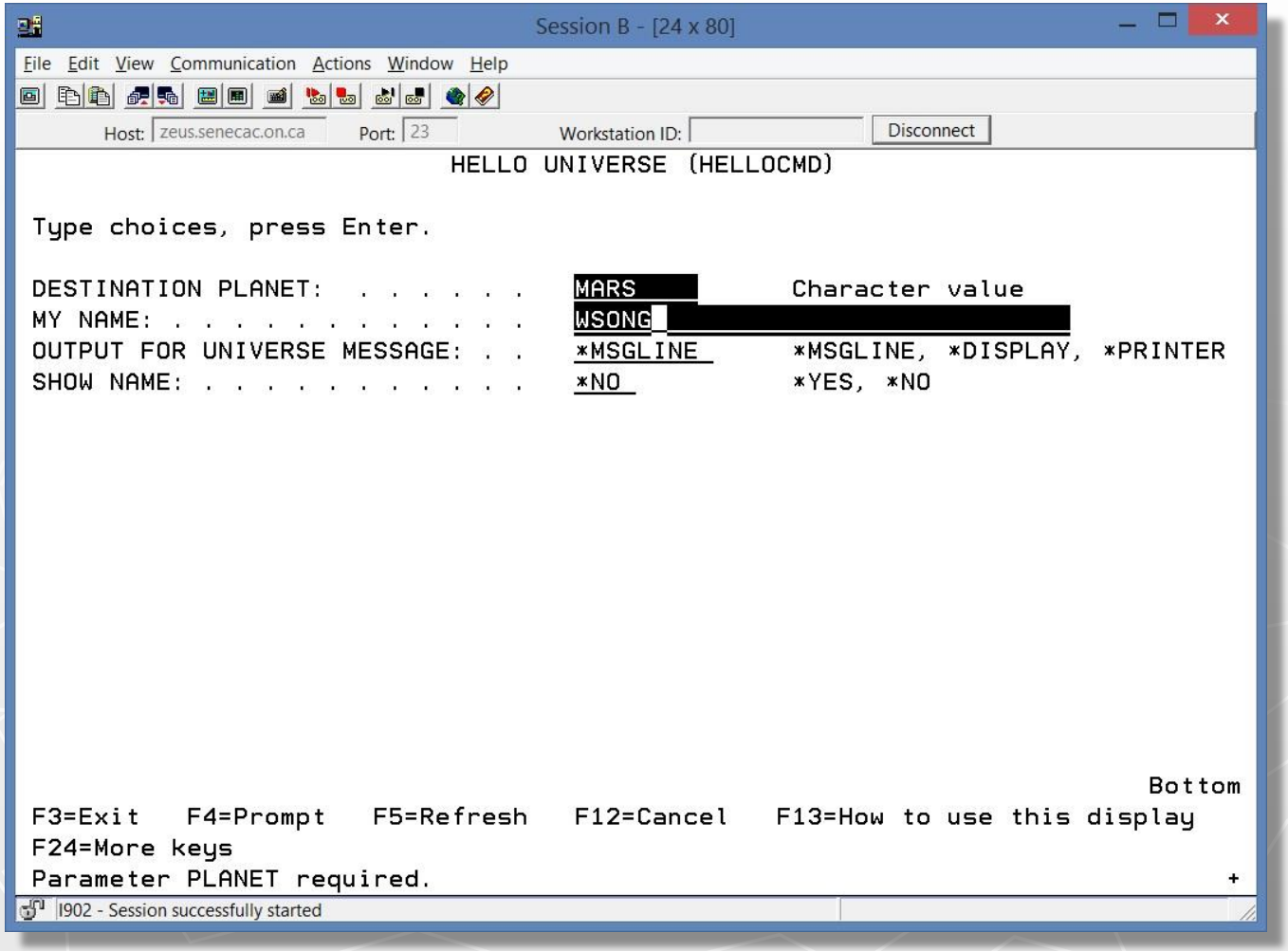
```
A          R RECORD1
A                                      4 22'H e l l o    U n i v e r s e'
A            PLANET         8   O      7 39
A                                      7 18'Destination Planet:'
A  99                                  9 18'Passenger Name:'
A  99        FULLNAME      31   O      9 39
```

- Create the command by entering the following command:

CRTCMD CMD(HELLOCMD) PGM(HELLOCPP) SRCFILE(QLAB8)

Note: QLAB8.file.ph-src is where your put your source files

Run the Command



F-SPEC Review

- File name – name of file
- File Type - C for a Display file
  I, U, or O for Database Objects
  O, for Printer Files (reports)
- File Designation – F for Full procedural file
- File Format - E for Externally Described
- Record Address Type - K if the object has a sort
- Device - Disk for Database Object
  Workstn for display files
  Printer for reports

RPG Verbs and Functions

- Read filename;
  - reads a record from a database object
- %EOF(filename)
  - Checks for End of File

Reference for display files

- Reference fields in an display file – the fields that can be defined by referring to the fields specified in a previously created database/pf file.
  - The field attributes referred to are the length, data type, and decimal positions of the field.
- You can specify R in this position 29 in order to use the reference function.
- Packed decimal and binary fields are not supported for display files. Therefore,
  - when you refer to fields of these types, the data type assigned is zoned decimal with a keyboard shift as follows: Y in position 35

Example of Reference Fields

```
.....AAN01N02N03..Name++++++RLen++TDpBLinPosFunctions++++++++++++++++++++++++++++++
     A                                      DSPSIZ(24 80 *DS3)
     A                                      REF(DS233C36/STUDENTS STUDENTSR)
     A          R RECORD1
     A                                      CF03(03)
     A            STUDENTNO R        O  4 18
     A            BIRTHDATE R        O  8 18
     A            PHONE     R     Y  O  4 46EDTWRD('0(   )&   -    ')
     A            ADDRESS   R        O 10 18
     A            CITY      R        O 11 18
     A            PROVINCE  R        O 11 35
     A            POSTALCODER        O 12 18
     A            FEESOWED  R     Y  O 16 18EDTCDE(1)
     A  90                                  DSPATR(RI)
     A            FINESOWED R     Y  O 14 18EDTCDE(1)
     A  91                                  DSPATR(RI)
     A                                   4  2'Student Number:'
```

The Data/base file

- STUDENTS.PF data file was defined in Lab 2.

Thank You!