**Lab objectives:**

**- Pass a parameter to an RPGLE program**

**- Provide file overrides to produce different report results**

**- Create a CLLE module that passes a parameter to an RPGLE Module and combine them into a**

**running program.**

**- Investigate the CHKOBJ command**

**Requirements to pass the lab:**

**Successfully run the Payroll program using a static call comprised of a CLLE driver module and a RPGLE module with various file overrides in effect that produces a multi page report. Make sure you know how to prove your running program has two modules. Your finished program should perform similar to the sample in BCI433LIB which can be run with the following commands:**

CHGCURLIB BCI433LIB (remember to change your current library back)

CALL PAYROLLW21

After your CLLE program finishes running have a message appear on the message line:

Okay to clear out an output queue in school - Never at work in production.

## RPGLE

#### Passing a Parameter

In Lab 5 we learned that the program that processes the SHIFTWEEK file could be run with a file override in effect to only process workers from a single shift. We are going to allow the report title some flexibility by passing the type of shift being reported on to the program. The report title will be passed as a parameter and will indicate Day Shift, Night Shift, Afternoon Shift or All Shifts.

In order to support this we need to make changes to the RPGLE program and to the Printer File.

Copy your Working RPGLE program from Lab 5 and change the name to PAYROLLPG2.

Copy your printer file from Lab 5 called PAYRPT and change the name to PAYRPT2

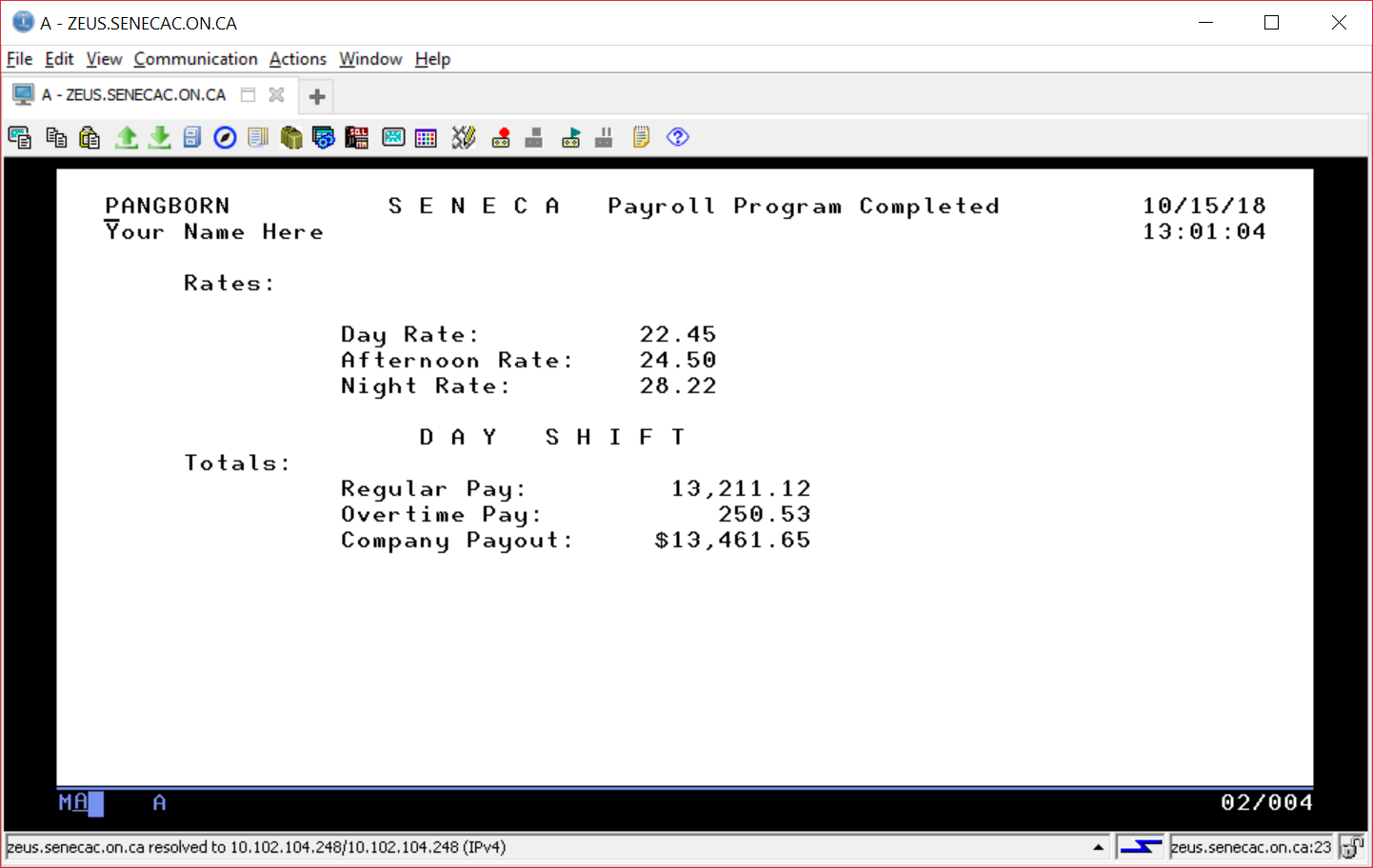
Copy your display file code from Lab 5 and change the name to PAYSUMMAR2

In PAYRPT2

On line 2 of your TITLE record include a named 30-character field in the centre of the line and call it SHIFTTYPE.

Compile your PRTF file member.

In PAYSUMMAR2 add the 30-character output only field of SHIFTTYPE so there is an indication if we are seeing totals from day, night, afternoon or all shifts. In the example below the total is for the day shift.



In PAYROLLPG2 make sure you support the changed display file and printer file names and add the following code to support passing a parameter:

Dcl-PI Main ExtPgm('PAYROLLPG2'); 🡨 This matches your program name

ShiftType char(30);

End-PI;

Compile your RPGLE program and try running it from the command line as:

==>CALL PAYROLLPG2 PARM(‘D A Y S H I F T’) .

The report title should now reflect the contents of the passed parameter.

#### File Overrides

In Lab 5 we created a view that only showed workers from the night shift.

We were unable to process workers from the night shift in employee number order. We could use SQL to provide an index object that provides the workers in employee number order, but we would not be able to select out just the night shift workers.

A view is a logical file on the power system. An index is also a logical file on the system. Although SQL does not provide for an object that can be a view and an index at the same time, a logical file coded with DDS can provide that type of object.

You are already familiar with the coding of a logical file, so the following replacement for it should make sense:

A R SHIFTWEEK PFILE(SENECAPAY/SHIFTWEEK)

A K EMPNUM

A S WORKSHIFT COMP(EQ '3')

You can run the command DLTF NIGHTS from the command line to remove the view created in Lab 5 and then produce the logical file object that processes only the night shift workers in employee number order.

Provide similar additional logical files for the other shifts (DAYS and AFTERNOONS)

#### CLLE Driver Program

We want to provide a CLLE program that asks the user what shift should be reported on and then produces the correct overrides and passes the appropriate report title when invoking the RPGLE payroll program.

This CLLE program can also override the printer file name so it reflects the shift type and can automatically display the spooled file on the screen.

Finally this CLLE program will be made flexible so it can use the output queue of the signed in user that is running it.

Here are the pieces for the CLLE program that you need to put together:

Interacting with the user without having to code a display file:

SNDUSRMSG MSG('1 - Day Shift, 2 - Afternoon Shift, 3 - Night Shift, 4 - All +

Shifts, 5 - EXIT') MSGRPY(&SHIFT)

The user can type a response that will be stored in the variable &SHIFT.

Changing your spool file name from PAYRPT2 to DAYSHIFT

OVRPRTF FILE(PAYRPT2) SPLFNAME(DAYSHIFT)

Looking at your last DAYSHIFT spooled file:

DSPSPLF FILE(DAYSHIFT) SPLNBR(\*LAST)

Table

Description automatically generated

Finding out the logged on user’s output queue and output queue library

RTVUSRPRF (you can press F4 to figure out the rest)

Removing the logged on users spooled files in their output queue

CLROUTQ

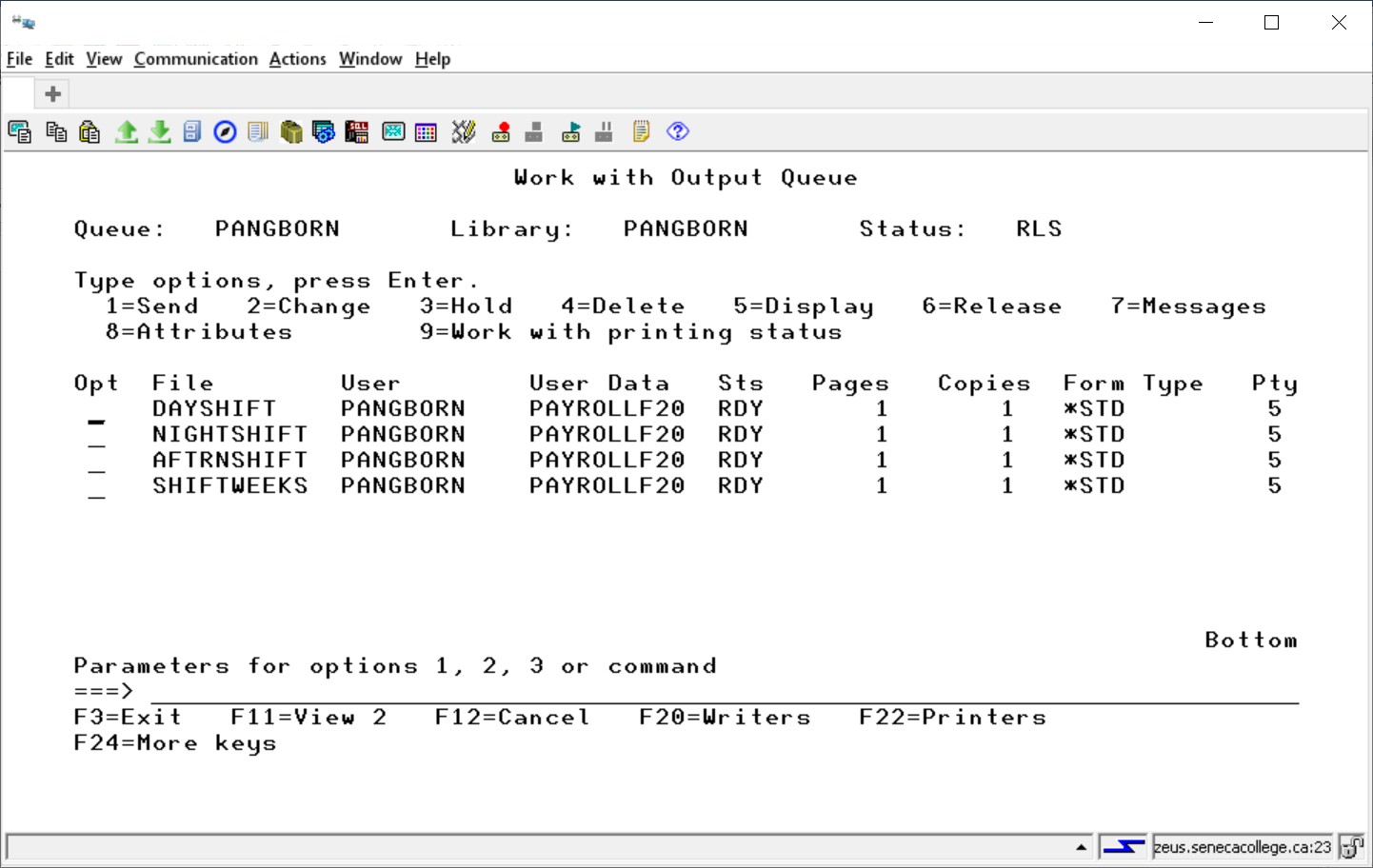
You already know how to invoke PAYROLLPG2.

To see how your driver program should run, add BCI433LIB to your library list and enter the following at the command line:

==> CALL RUNPAYPGM2

Test it out by entering a “1”, “2’, “3’, “4”, 6”, Enter key and a “5”

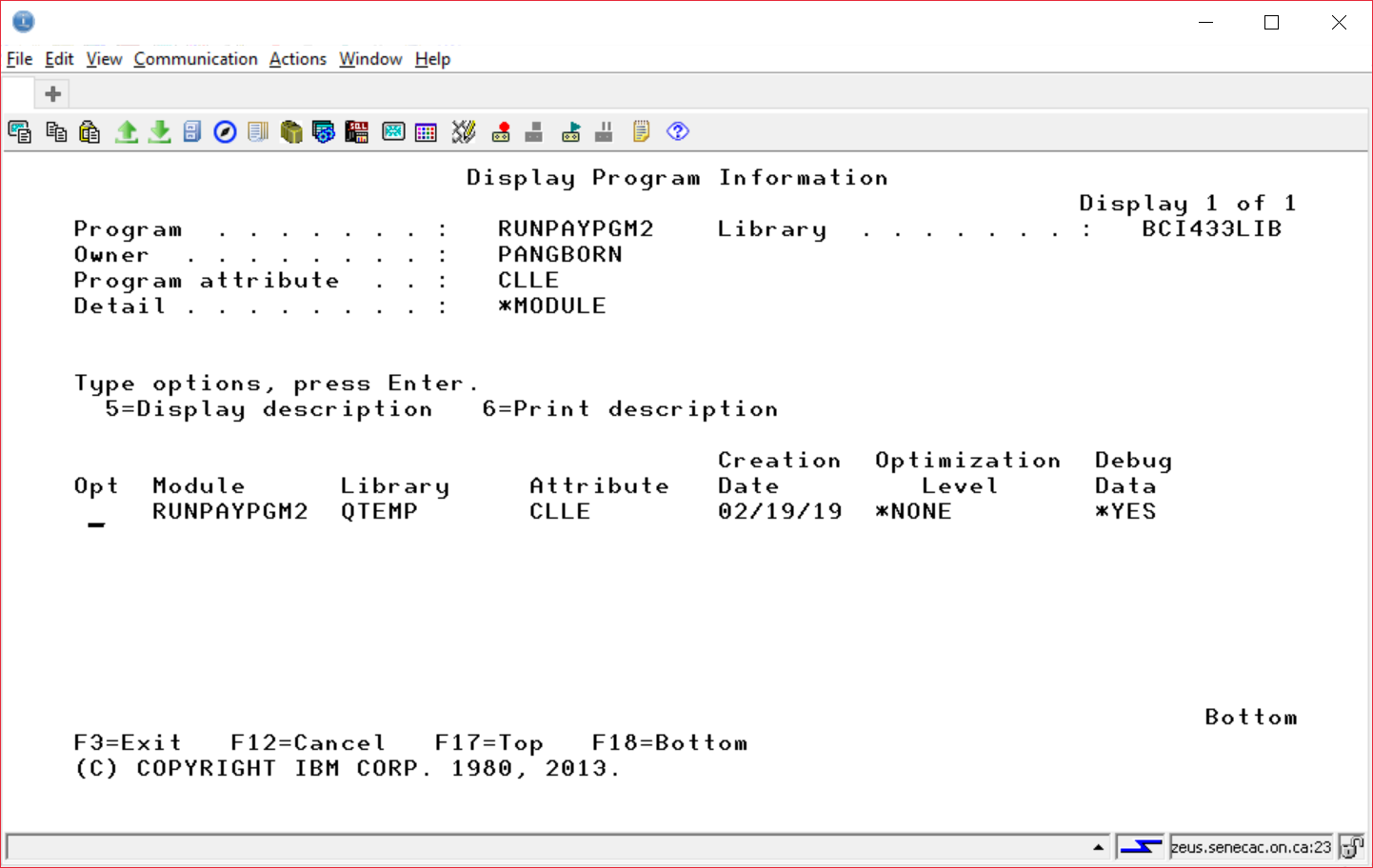
Your program should finish with a screen similar to the one shown below



The code for RUNPAYPGM2 will be discussed in class. If you miss this class, you will need to get notes from someone who was there. After finishing this CLLE program called RUNPAYPGM2 and successfully running it, we will copy it and rename it to RUNPAYPGM3.

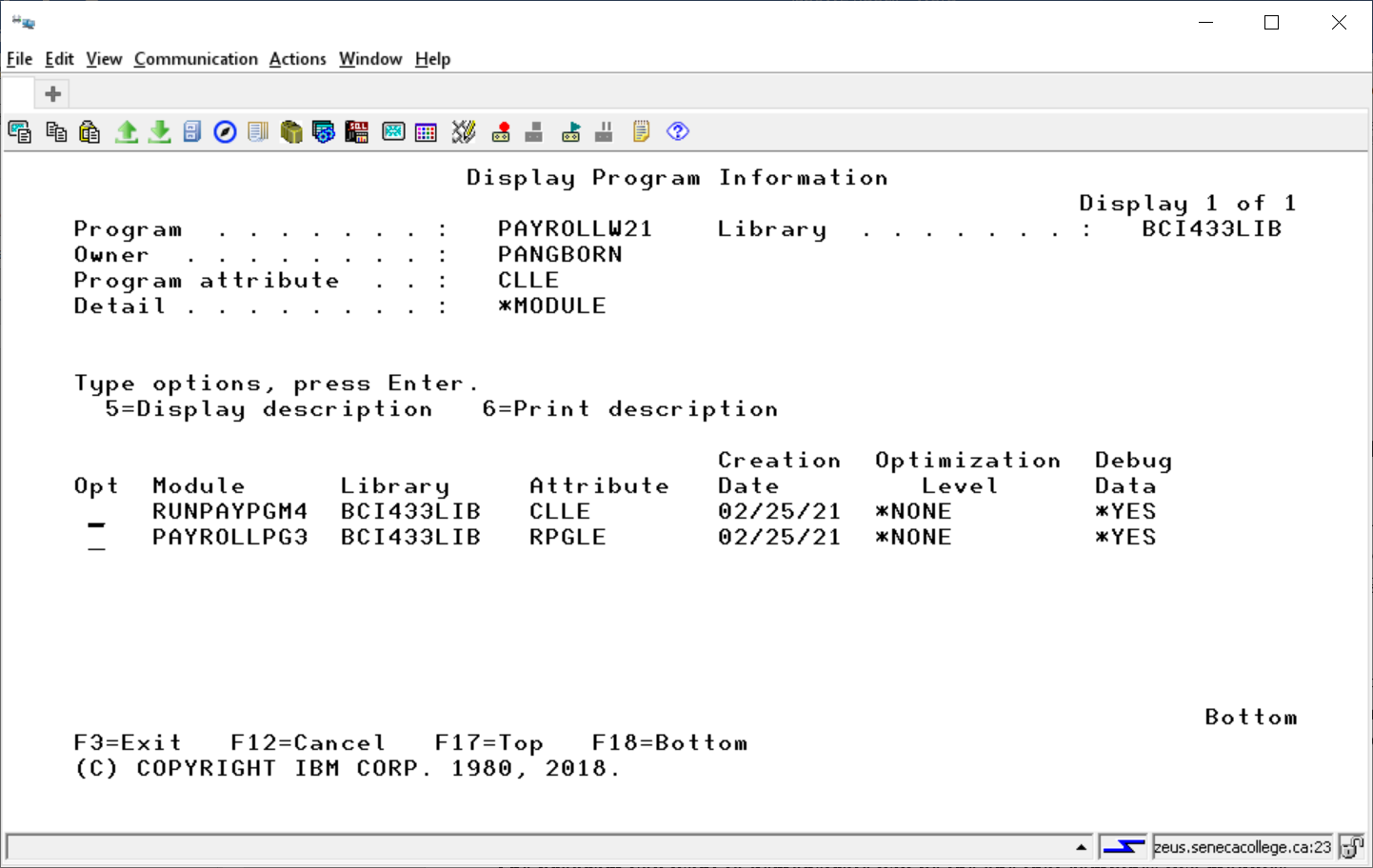
Lets see the programs available in BCI433LIB.

DSPPGM PGM(BCI433LIB/RUNPAYPGM2) DETAIL(\*MODULE)



DSPPGM PGM(BCI433LIB/PAYROLLW21) DETAIL(\*MODULE)

(note your version would refer to RUNPAYPGM3 rather than RUNPAYPGM4)



If you compare RUNPAYPGM2 to PAYROLL you will see that the first one has a single module and the second one has two modules.

RUNPAYPGM2 and PAYROLLPG2 were created with CRTBND? commands and a CALL statement is used to invoke the RPGLE program. That means at run time extra work is required to link the two programs and provide support for the parameter passing. This must occur every time the RUNPAYPGM2 program is run.

This is referred to as a dynamic call.

RUNPAYPGM3 and PAYROLLPG3 were created with CRT???MOD commands and a CALLPRC statement is used to invoke the RPGLE program module. The linking of the two programs and resolving addresses etc can now take place at a CRTPGM step done from the command line a single time. It does not occur every time the PAYROLL program is run.

This is referred to as a static call.

**The program you want to demonstrate will be the one that combines two modules.**

The steps to produce the two modules are selecting the following compile commands in RDi

CRTCLMOD – for the CLLE program

CRTRPGMOD – for the RPGLE program

Combining the modules would involve the following one time command:

===> CRTPGM PGM(PAYROLL) MODULE(RUNPAYPGM3 PAYROLLPG3)

You should be aware that any time you change a module’s code and recompile it, the old module with the problem code is still in the program object. You need to rerun the CRTPGM command again to replace the faulty module.

Notes:

1. You could have the same name for the first parameter as the calling module PGM(RUNPAYPGM3) In this case you would run your program with CALL RUNPAYPGM3 instead of PAYROLL.

2. RUNPAYPGM3 has to be specified first since it is the program that calls PAYROLLPG3. You can not have typed MODULE(PAYROLLPG3 RUNPAYPGM3) for the second parameter of the CRTPGM command.

A Review of Essential Concepts

\*PGM VS \*MODULE

CRTBNDRPG VS CRTRPGMOD

CRTBNDCL VS CRTCLMOD

Modules combined with CRTPGM producing a \*PGM object

DYNAMIC CALL VS STATIC CALL

CALL VS CALLB (CALL BOUND) an RPGLE statement

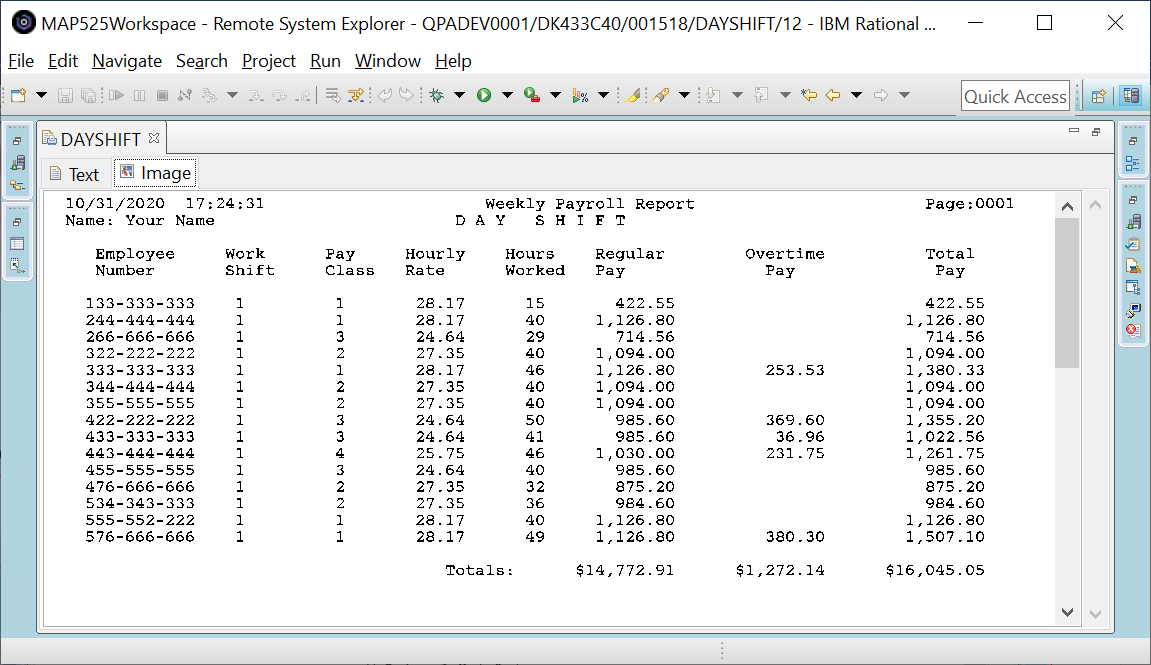
CALL VS CALLPRC (CALL PROCEDURE) A CLLE statement

OVERRIDES

OVRDBF

OVRPRTF

Make sure you have similar spacing when viewing in RDi



Investigating commands in Client Access:

Sometimes CLLE driver programs will check and see if a required object exists and the user profile currently signed in has access rights to the object.

Type CHKOBJ (press F4) (press F1 for parameter help) (Press F2 for extended help)

The Check Object (CHKOBJ) command checks object existence and verifies the user's authority for the object before trying to access it. If the object exists and the user has the proper authority for the object, no error messages are sent to the user.

What message to you get when you type the following at the command line.

CHKOBJ OBJ(BCI433LIB) OBJTYPE(\*LIB) AUT(\*CHANGE)

Not authorized to object BCI433LIB in QSYS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You could monitor for that message in a CLLE program.

Put your cursor on the error message and press F1. What message ID would you monitor for? CPF9802

 **BCI433**

LAB 5

PAYSUMMARY.\*file.dspf

PAYRPT1

.\*file.prtf

PAYROLLPGM

.\*pgm.rpgle

**LAB 6**

SENECAPAY/ SHIFTRATES

.\*file-pf-da

DDS

view: NIGHTS

.\*file.lf

DDS

view: DAYS

.\*file.lf

DDS

view: AFTS

.\*file.lf

PAYSUMMAR2.\*file.dspf

PAYRPT2

.\*file.prtf

SENECAPAY/ SHIFTWEEK

.\*file.pf-dta

PAYROLLPG2

.\*pgm.rpgle

static call

PAYROLL

.\*pgm.clle

RUNPAYPGM3

.\*module (CLLE)

PAYROLLPG3

.\*module (RPGLE)

SENECAPAY/ SHIFTRATES

.\*file-pf-da

SQL view: NIGHTS

.\*file.lf

SENECAPAY/ SHIFTWEEK

.\*file.pf-dta

|  |  |  |
| --- | --- | --- |
| RUNPAYPGM2  .\*pgm.clle | | |
| dynamic call |  | |
|  | |  |