
EXERCISE 1

Write and Read Miracle Analysis Header File

```
$ Miracle Analysis Input Deck, PDA Institute training, class  
PAT305  
$ File: /patran/patran3/template_Miracle.db  
$ Date: 16-Dec-93  
$ Time: 15:10:31  
$ Jobname : clevis  
$ Job Description : Miracle job created on 10-Sep-93 at 10:41:13  
*Job control  
>Loadcase = Default  
>Convergence Tolerance = 0.
```

Objectives:

- Write two PCL functions which write and read a file.
- Compile the functions
- Verify that the function works



Problem Description:

In this Exercise we write two PCL functions. The first function, **miracle_write_job_header()**, writes the job information to a fictitious analysis code named Miracle. The second function, **e01_read_text()**, Reads the first line of the file.

Suggested Exercise Steps:

- Write two PCL functions using the on-line editor.
- Compile the function
- Verify the PCL functions.

Exercise Procedure:

Write a PCL Function

1. Either use vi or jot as the text editing tool. Open a file named **miracle_write_job_header.pcl**.
 - If you have used vi before and would like to refer to the QuickReference card, turn to the back of your exercise book and look at Appendix A.

Write two functions as follows:

```
FUNCTION miracle_write_job_header( dbname, analysis_type, @
                                job_name, job_desc, load_case, @
                                convergence_toler )

/*
 * Write the analysis header information to the input deck for
 * the Miracle Analysis code
 *
 * INPUT:
 * dbname          STRING Database name
 * analysis_type   STRING Analysis type
 * job_name        STRING Current job name
 * job_desc        STRING Current job description
 * load_case       STRING load case name
 * convergence_toler REAL Convergence tolerance
 *
```

```

* Side Effects:
* The file job_name.inp is created and the date, time, and
* database name are added as comments (Line starts with $).
* The analysis information for "job_name" are written to the
* input deck.
* job_name.inp should look something like this:
*
* $ Miracle Analysis Input Deck, PDA Institute training, class
* $                               PAT305
* $ File: database_name
* $ Date: 14-Jun-93
* $ Time: 10:33:46
* $ Jobname : job_1
* $ Job Description : This is a short description of the job
* $ Job control
* >Loadcase = load_1
* >Convergence Tolerance = 0.0049999999
*
*
* Errors:
* Return 0, no error
* Otherwise it is a file utility error
*/
(fill in)
END FUNCTION
FUNCTION e01_read_text( filename, text )
/ *
* Read a string of text from a file.
* INPUT:
*   filename      STRING   Filename Prefix. ".txt" is
*                               automatically appended
* OUTPUT:
*   text          STRING   String to write to file
*/
(Fill in)
END FUNCTION

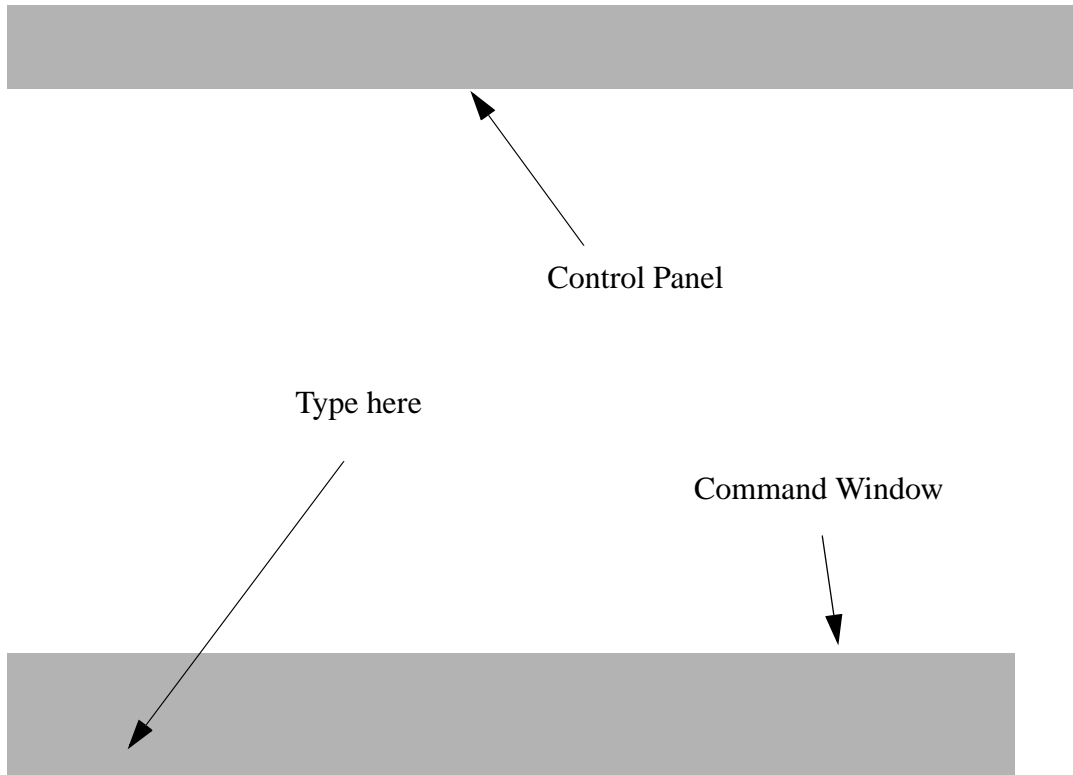
```

2. Compile the functions.

Type **p3** at the prompt and **<return>**.

Exercise 1

After the main menu and command window appear, type
!!input miracle_write_job_header.pcl in the command
line:



Resolve any compile errors by editing the PCL function
and re-compiling.

3. Test the function.

In the command line type:

```
miracle_write_job_header("database_name","structural","job_1",@  
                        "This is a test","load_1",.005)
```

Type **<return>** and:

in the command line type:

```
string text[132]  
e01_read_text( "job_1", text )
```

in the command line.

4. Verify the result:

Look to see if the first line of the file "job_1.inp" has been read back into the PCL variable **text**.

Type:

```
dump text
```

p3 should respond with:

```
$# STRING text[132] = "$ Miracle Analysis Input Deck, PDA Institute  
training" // @  
$# ", class PAT305"
```



Exercise 1



Sample Solution

```

FUNCTION miracle_write_job_header( dbname, analysis_type, @
                                job_name, job_desc, load_case, @
                                convergence_tolер )

/*
* Write the analysis header information to the input deck for
* the Miracle Analysis code
*
* INPUT:
* dbname          STRING Database name
* analysis_type   STRING Analysis type
* job_name        STRING Current job name
* job_desc        STRING Current job description
* load_case       STRING load case name
* convergence_tolер REAL Convergence tolerance
*
* Side Effects:
* The file job_name.inp is created and the date, time, and
* database name are added as comments (Line starts with $).
* The analysis information for "job_name" are written to the
* input deck.
* job_name.inp should look something like this:
*
* $ Miracle Analysis Input Deck, PDA Institute training, class
* $                               PAT305
* $ File: database_name
* $ Date: 14-Jun-93
* $ Time: 10:33:46
* $ Jobname : job_1
* $ Job Description : This is a short description of the job
* *Job control
* >Loadcase = load_1
* >Convergence Tolerance = 0.0049999999
*
*
* Errors:
* Return 0, no error
* Otherwise it is a file utility error
*/

```

Exercise 1

```
STRING dbname[], analysis_type[], job_name[], job_desc[]
STRING load_case[]
REAL convergence_tolер
INTEGER status, fid
file_delete( job_name//".inp" )
status = text_open( job_name//".inp", "NRW", 0, 0, fid )

IF( status != 0 ) THEN
write("Unable to open file "//job_name//".inp")
RETURN status
END IF

/*
* Put the date and time stamp in the file
*/
text_write_string( fid, "$ Miracle Analysis Input Deck, //@
                  " PDA Institute training, class PAT305")
text_write_string( fid, "$ File: "//dbname )
text_write_string( fid, "$ Date: "//sys_date() )
text_write_string( fid, "$ Time: "//sys_time() )
text_write_string( fid, "$ Jobname : "// job_name )
text_write_string( fid, "$ Job Description : "// job_desc )
text_write_string( fid, "*Job control" )
text_write_string( fid, ">Loadcase = "//load_case )
text_write_string( fid, ">Convergence Tolerance = "// @
                  str_from_real( convergence_tolер ) )

/*
* Close the external file
*/

text_close( fid, " " )

RETURN 0

END FUNCTION
```

```
FUNCTION e01_read_text( filename, text )
/ *
* Read a string of text from a file.
* INPUT:
*   filename   STRING   Filename Prefix. ".txt" is
*                                   automatically appended
* OUTPUT:
*   text       STRING   String to write to file
*/

STRING text[], filename[]

INTEGER status, fid, lrecl

status = text_open( filename//".inp", "RO", 0, 0, fid )

IF( status != 0 ) THEN
write("Unable to open file "//filename//".inp")
RETURN status
END IF

/*
* Read the string
*/

status = text_read_string( fid, text, lrecl )

/*
* Close the external file
*/

text_close( fid, " " )

RETURN 0

END FUNCTION
```