

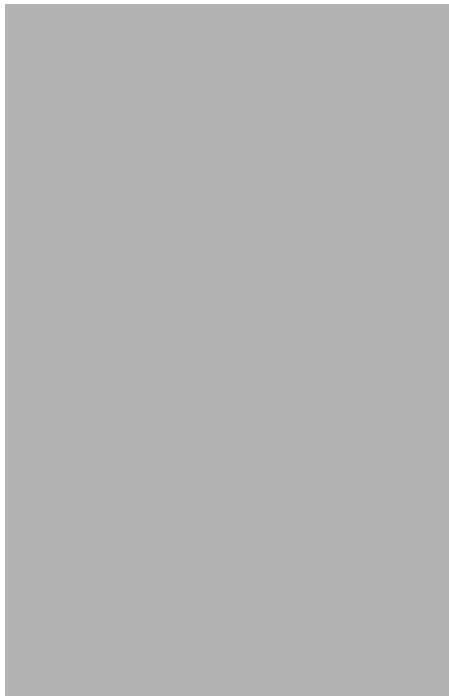
---



## EXERCISE 12



# *Spawning the Analysis Job*



### **Objectives:**

- Add spawning of both the forward and reverse translator to the Apply callback
- Add spawning the actual analysis



## Problem Description:

In this Exercise we edit the PCL function **miracle\_analysis\_ex12.template** to spawn the forward translator, reverse translator and analysis job.

As described in the lecture, you are provided with the Forward\_translator, analysis code, and the reverse\_translator for the Miracle program. In this exercise, you will write appropriate “pcl” functions to spawn Pat3Mir, Miracle, and MirPat3 programs. The forward translator, **pat3mir** executes as follows:

```
%pat3mir database_name miracle_input_filename
example: pat3mir test.db test.inp
```

The analysis code, **Miracle**, executes like:

```
Miracle jobname
example: Miracle test
```

The analysis code **Miracle** generates three files as follows:

- *job\_name.dis*

Contains nodal displacements in the standard PATRAN2.5 neutral format.

- *job\_name.els*

Contains element stress results in the standard PATRAN2.5 neutral format.

- *job\_name.ans*

Contains two lines which are the file names for the .dis and .els file for the current job.

- The reverse translator is called **mirpat3**. It requires a file as input, which contains three lines, named **miracle\_res\_file.txt**

The following lines must be written to the **miracle\_res\_file.txt** file:

```
template file
Results file (PATRAN2.5 format)
Database name
Date
```

The template file is of the same format as the PATRAN 2.5 template files for P3. These files are provided and are located in your directory:

```
miracle_res_file.dis_tmpl
miracle_res_file.els_tmpl
```

---

Use `miracle_res_file.dis_tmpl` to read displacement results and `miracle_res_file.els_tmpl` to read element results.

## Suggested Exercise Steps:

- Edit `miracle_analysis_ex12.template` to add `utl_process_spawn()` calls..
- Compile `miracle_analysis.pcl` into **miracle.plb**
- Edit `miracle_reverse_translate.template` to add `utl_process_spawn()` calls..
- Compile into `miracle_reverse_translate.pcl` **miracle.plb**
- Build the supplied forward-,reverse-translator and analysis code for “**Miracle**”.

## Exercise Procedure:

1. Either use `vi` or `jot` as the text editing tool. Edit the two aforementioned files and supply the missing PCL statements. The first function contains a class called `apply` and the other is a standard PCL function called by the former.
2. Compile **miracle\_analysis.pcl** and **miracle\_reverse\_translate.pcl** into **miracle.plb**.

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

After the main menu and command window appear, type:

```
!!compile miracle_analysis.pcl into miracle.plb
!!compile miracle_reverse_translate.pcl into miracle.plb
```

Resolve any compile errors by editing the PCL functions and re-executing the compile command.

3. Select Quit from the File menu..
4. Build the supplied forward-,reverse-translator and analysis code for “**Miracle**”.







## Sample Solution

```

/*$$h ANALYSIS Miracle Analyze Job Function */
/*
* Purpose:
* Get job control data and spawn job for a miracle analysis
*
* Input:
* <none>
*
* Output:
* <none>
*
* Side Effects:
* A miracle analysis job is spawned.
*
*/

CLASS miracle_analysis

CLASSWIDE INTEGER cur_job_id, cur_param_set_id, cur_job_status
CLASSWIDE STRING cur_job_desc[256], cur_job_name[80]

/*$$ FUNCTION apply
* Purpose:
* To retrieve all pertinent widget data, write the job data to db
* and initiate the job submit procedure, if required.
*
* Input:
* <none>
*
* Output:
* <none>
*
* Action:
* Creates the jobfile, initiates the job submit procedure
*
*/

FUNCTION apply

GLOBAL REAL convergence_tolcr

STRING analysis_code[32], dbname[132], @
labels[80](3), analysis_type[80]
STRING steps[80](1)
STRING answer_file[200], record[80], type[3]
STRING object_val[32],filespec[256]

WIDGET lbox_id

INTEGER db_get_status, positions(3), analysis_code_id, analysis_type_id, @
status, chan, lrecl, res_chan

/*
* Get the name of the currently open database
*/

db_get_status = trans_get_db_name( dbname )

```

## Exercise 12

---

```
/*
 * Get the current analysis code, id and type of analysis
 */

analysis_main.get_code_and_type( analysis_code, analysis_code_id, @
analysis_type, analysis_type_id )

/*
 * Get current job info and menu items
 */

analysis_main.get_current_job_info ( cur_job_name, cur_job_desc, @
cur_job_id, cur_param_set_id, @
cur_job_status )

analysis_main.get_analysis_menu_items( positions, labels )

/*
 * For "Read Results" action branch according to object
 */

IF( labels(1) == "Read Results File" ) THEN

/*
 * Get the result file names from the .ans file
 */

analysis_select_file.get_res_file_name( answer_file )
> miracle_reverse_translate( answer_file, dbname )
ELSE

/***** job information *****/

/*
 * Create a title if there is none
 */

IF( STR_LENGTH( cur_job_desc ) == 0 ) THEN
padvancedfea_job_tasks.create_job_title( analysis_code, @
analysis_type, cur_job_name, cur_job_desc )
END IF

/*****Load Case Selection*****/

/*
 * Get the selected load cases
 */

miracle_loadcases.get_lc_widget( lbox_id )

ui_wid_get( lbox_id, "VALUE", steps(1) )

dump steps(1)

IF( steps(1) == "" ) THEN
write("No job selected. Analysis aborted")
RETURN -1
END IF
```

```

/*****Submit Job *****/

/*
 * Spawn the analysis
 */

/*
 * Write the header information to the input deck
 */

> status = miracle_write_job_header( dbname, analysis_type, cur_job_name, @
cur_job_desc, steps(1), convergence_tol )
IF( status != 0 ) THEN
msg_to_form( status, 4, appcode(status), 0, 0.0, "" )
RETURN
END IF

/*
 * Spawn the job
 */

> ***** 1 *****
IF( status != 0 ) THEN
msg_to_form( status, 4, appcode(status), 0, 0.0, "" )
RETURN
END IF

> ***** 2 *****
IF( status != 0 ) THEN
msg_to_form( status, 4, appcode(status), 0, 0.0, "" )
RETURN
END IF

/*
 * Run the forward translator
 */

status = file_get_filespec( "pat3mir", "OP", filespec )

***** 3 *****
> ***** 3 *****/@
***** 3 *****

IF( utl_process_error( status ) ) THEN
utl_display_process_error( status, 2 )
> status_add( "pat3mir", dbname, "Translation Failed" )
RETURN -1
ELSE
> status_add( "pat3mir", dbname, "Translation Completed" )
write_line("Forward Translation Complete for "//cur_job_name)
END IF
> ***** 4 *****

/*
 * Run the analysis
 */

status = file_get_filespec( "Miracle", "OP", filespec )
IF( status != 0 ) THEN
write( "Failed to find file Miracle" )
RETURN -1
END IF
> ***** 5 *****

```

## Exercise 12

```
> ***** 5 *****

  IF( utl_process_error( status ) ) THEN
> utl_display_process_error( status, 2 )
> status_add( "miracle", dbname, "Analysis Failed" )
  ELSE
> status_add( "miracle", dbname, "Analysis Completed" )
> write_line("Analysis completed successfully for "//cur_job_name)
  END IF

END IF /* for labels(1) */

END FUNCTION /* apply */

END CLASS /* miracle_analysis */

/*$$ FUNCTION miracle_reverse_translate
* Purpose:
* to run the reverse translation of a miracle analysis and to
* read the results into the current database
*
* Input:
* answer_file STRING file which contains the list of files to read
* dbname STRING Database file name
*
* Output:
* <none>
*
*/

FUNCTION miracle_reverse_translate( answer_file, dbname )

STRING answer_file[]
STRING dbname[]

STRING record[80], type[3]

INTEGER status, chan, lrecl, res_chan

text_open( answer_file, "OR", 0,0, res_chan )

/*
* Only one object which is "Result Entities"
* Begin by writing the file which controls the reverse
* translation for nodal displacements (.dis file)
*/

uil_file_close.go()

WHILE ( text_read_string( res_chan, record, lrecl ) == 0 ) loop

/*
* Check for each file type
*/

type = ""
type = str_token( record, ".", 2, TRUE )
```

```

/*
 * Write the reverse translator file driver:
 * "miracle_res_file.txt"
 */

IF( file_exists( "miracle_res_file.txt","") ) THEN
IF( ui_read_logical("File miracle_res_file.txt exists,"//@
" do you want to overwrite it?")) THEN
file_delete( "miracle_res_file.txt" )
ELSE
text_close( res_chan, "" )
uil_file_open.go( dbname )
RETURN
END IF
END IF

status = text_open( "miracle_res_file.txt", "NW", 0,0, chan )
text_write_string( ***** 1 ***** )
text_write_string( ***** 1 ***** )
text_write_string( ***** 1 ***** )
text_write_string( chan, "Date: "//sys_date()//" "//sys_time() )

text_close(chan,"")
status_add( "mirpat3", dbname, "Starting reverse "//@
"translation")
status = utl_process_spawn(**** 2 ***** )
IF( utl_process_error( status )) THEN
write( "mirpat3 failed")
status_add( "mirpat3", dbname, "Reverse "//@
"translation failed")
text_close( res_chan, "" )
uil_file_open.go( dbname )
RETURN
ELSE
status_add( "mirpat3", dbname, "Reverse "//@
"translation completed successfully")
END IF
text_close( chan, "" )
file_delete( "miracle_res_file.txt" )
END WHILE
text_close( res_chan, "" )
uil_file_open.go( dbname )

END FUNCTION /* miracle_reverse_translate */

```

---

## Solutions

```

1)status = loadprocs_eval_all()
2)status = db_commit_raw()
3)status_add( "pat3mir", dbname, "Starting translation" )
   status = utl_process_spawn( "//dbname"//@
   cur_job_name//".inp", TRUE )
4)db_start_transaction_raw()
5)dd
   status = utl_process_spawn( filespec//" "//cur_job_name, TRUE )

```



**Exercise 12**



---

