EXPERIMENT SCRIPT FOR THE SCENARIO II (GROUP B)

1. SUMMARY

The purpose of the experiment is to analyze the manual feature transfer process between two systems, the donor and the receiver (host), both written in C. For this process, we will record the necessary activities, time, and effort spent by a set of SPL specialists compared to our proposed automated approach. The process consists of four steps: extraction, identification, adaptation, and insertion.

The participants will have access to a set of unit tests, a feature's entry point in the donor (a function), and an insertion point in the host system provided by the researchers. Participants may use the tools they prefer to try to perform the process in the shortest time possible. We request that participants record all activities, steps, and time spent, as well as the tool they used.

It is important to note that the experiment will not require participants to perform the process without any breaks, as long as they record time and activities correctly each time they return to the execution process.

2. TRAINING

We have prepared some videos describing each section of this script. You can assist at any time while performing the experiment. We only ask you not to register this time in the time and effort spreadsheet.

3. EXPERIMENT EXECUTION ACTIVITIES

1. Download the experiment files, available at: experiment_directory;
2. Access experiment form containing the access link to your time and effort registration spreadsheet: registration_sheet;
3. Install dependencies;
4. Execute the Feature transferring process;
5. Execute the unit test;
6. Perform system testing in the post-transference receiver system;
7. Send files to researchers via the experiment form;
8. Signal the end of the experiment to researchers.

4. EXPERIMENT DIRECTORY

Execution Directory: As part of the experiment preparation process, the participant should download the files contained in the folder corresponding to the group in which he was allocated. All files needed for the experiment can be downloaded by the link: transplantation_directory

Effort Register Worksheet: For time and effort registration, we provide a spreadsheet for each participant. You can it by the link: effort_form

4.1. DIRECTORIAL STRUCTURE

GROUP B

| _ Dependencies_run -> Dependency Installation Files
| _ Documentation -> Contains the Donor and host system documentation if the participant wants to better understand the systems.
| _ Owner -> Contains the source code of the donor system.
| _ Host_original -> Contains the source source code of the receiving system (host)
| _ Host_to_transplant -> Contains the host source code to receive the feature
| _ Test_Suite -> Temporary directory for feature and unit tests
| _ Feature -> Temporary directory of the feature source code insertion

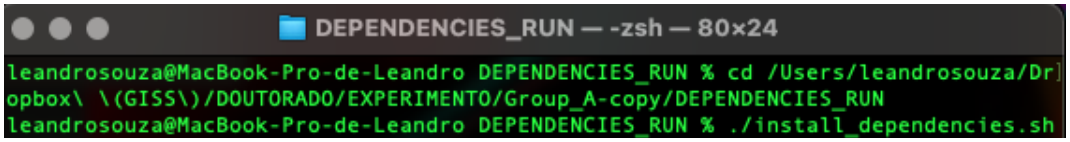
5. DEPENDENCIES INSTALATION

Please install some dependencies required to execute unit tests. **Do not worry about installing these premises, at the end you can run the script that will be automatically generated after the facilities in the directory will be installed Installation of dependencies (dependencies_run)**

We have prepared a shell script to install the dependencies automatically from the directory: Dependencies_Run execut

5.1. INSTALLATION OF DEPENDENCIES (FOR MAC OS)

From the directory Dependencies_Run execut the following command: `./install_dependencies.sh`



Terminal screen with CD commands e ./install_dependencies.sh

This command will install the following dependencies:

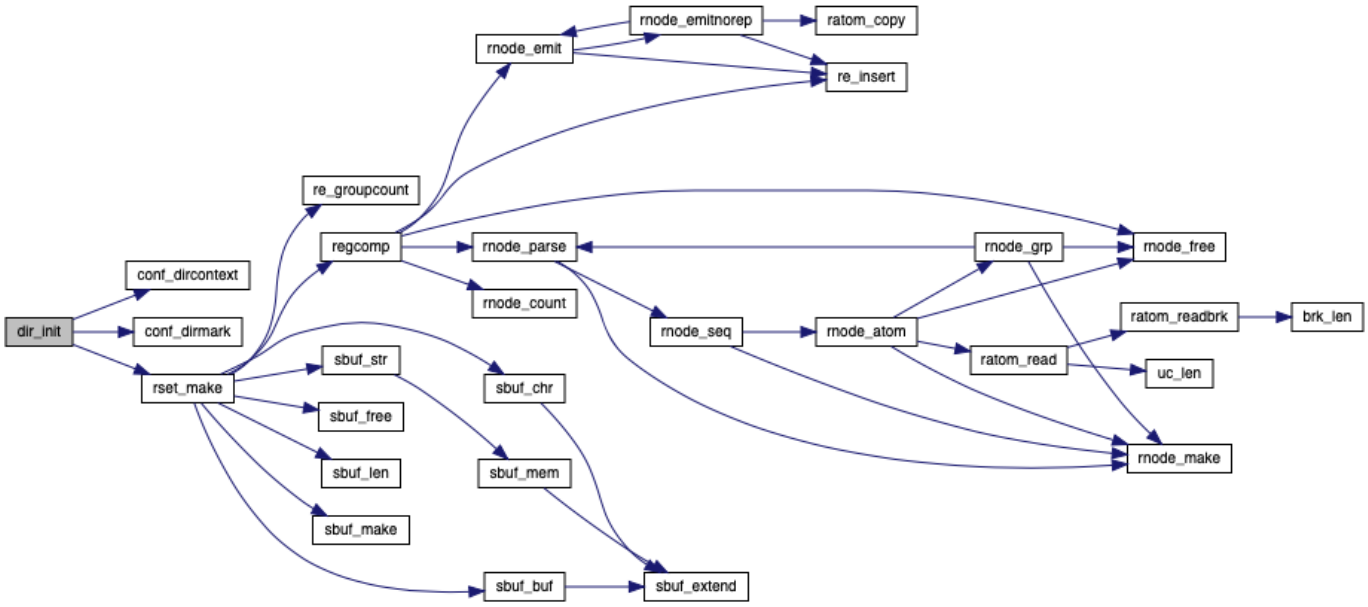
- Autoconf
- libtool
- Pkg-config
- Check
- Glib

6. DESCRIPTION OF SYSTEMS AND FEATURE

We divide the participants of our experiment into two groups (A and B). This script describes the process to be performed by group B participants. As a member of Group B, you will try to transfer the feature *dir_init* between two versions of the *Neatvi* text editor, *Neatvi_1.0* version (receiver system) and *Neatvi_2.0* (fetal donor). The feature *dir_init*, present only in the *Neatvi_2.0* version, allows the opening of a file already created. For this, the function performs changes in global variables:

```
struct rset *dir_rslr;    /* pattern of marks for left-to-right strings */
struct rset *dir_rsrl;    /* pattern of marks for right-to-left strings */
struct rset *dir_rsctx;    /* direction context patterns */
```

, present in the *dir.c*. Without such variables will be null but do not interfere with the functioning of the system. To function correctly it will be necessary to extract all functions called from the *dir_init ()* function implemented in the *dir.c* file, following its call chart.



6.1. AVAILABLE ARTIFACTS

As they are distinct systems it will be necessary to adapt the code that implements the entry point of the feature (ie the *dir_init()* function) for the insertion point in the *NEATVI* system, based on the following information:

- **Feature input point:** place where a call to the dir_init () function will be inserted, the entry point of the feature implemented in the dir.c file.
- **Host system insertion point:** point defined by notation with __ADDGRAFTHERE_JUSTHERE in file vi.c
- **Call_GRAPH:** The Feature call chart available both in your documentation in documentation/owner_documentation/index and can be viewed through the documentation/call_graph.png image.
- **Test_Suite:** A set of unit tests that can be used to assist in adapting and executing the feature before its insertion. Such test files are available in the transfer directory and have already been implemented by the researchers, and participants are only the task of executing them from the commands described in Section 6.

These artifacts are described more detailed in the following sections.

7. FEATURE TRANSPLANTATION PROCESS

In this section, we describe the systems and features involved in the transfer process and the step-by-step process for executing the feature transfer. We provide each participant with training mini-videos of the process that can be viewed as many times as necessary. If there are still any questions after reading the step-by-step instructions and viewing the training, please contact the researchers.

The purpose of this experiment is to analyse the manual feature transfer process between two versions of the same system written in C. You may use the IDE you feel comfortable with or just a simple text editor.
You should record the activities and time spent on the individual time collection spreadsheet provided. Consider any action taken during each stage of the process as an activity, for example, IDENTIFICATION STAGE: searching for global variables used by extracted functions. Do not count the time for training and reading these scripts. Below we highlight the stages of the process that will be counted.

7.1. PROCESS STEP

Once the premises are installed, you can start execution of the experiment. Have the activity and time -spent registration worksheet available.
We divide the experiment into 4 steps: extraction, identification, adaptation and insertion of feature. Next we describe each step.

We divide the experiment into 3 steps: extraction, adaptation and insertion of feature. Next we describe each step.

1. Extraction: corresponds to the extraction process of the entire portion of code that implements the feature, necessary for its execution (functions) from the function that determines the entry point of the feature and its call chart. All code implements the feature should be copied to the host_to_transplant directory, environment with the host system. If the function already exists in the receiving system, you should involve it with the F_DIR_INIT directive.

All copied elements must be delimited with FLAG F_DIR_INIT. As it is the transfer of features between two versions of the same system it will be common to find functions already implemented in the receiving version. In this case, you should delimit the function with FLAG F_DIR_INIT_TOO. Both flags are already defined in file *vi.h* (lines 4 and 5).

To facilitate this process, we provide the call graph. The full version of it is available in the Documentation directory.

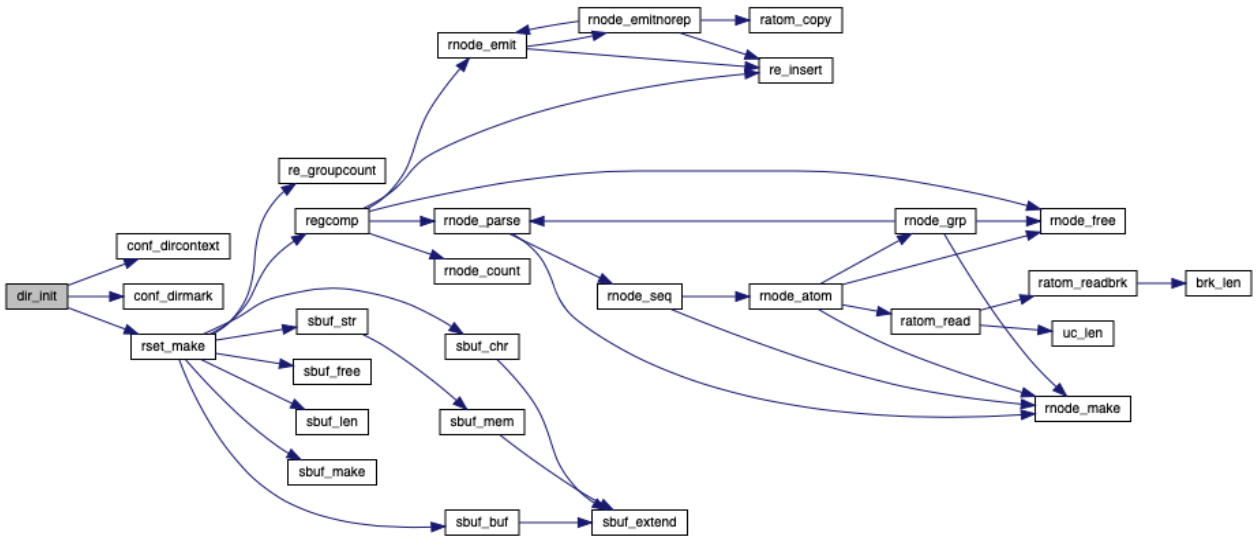


*Grafico de chamada da feature **dir_init***

2. **Adaptation**: You should, in addition to extracting the functions called by Feature, analyze whether any portion of receiving system code needs to be adjusted to load the parameters correctly from the feature point of the feature in the host environment. The inserting point is defined with the notation: *__ADDGRAFTHERE__JUSTHERE*, which can be found in the *main* function of file vi.c.

3.1 Test Execution: Once all these steps are done, the host system will be ready to perform the unit tests already implemented and made available in the **Test_Suite** Directory. These tests have already been implemented, requiring the participant only to execute the commands described in *Section 6* of this document. It is possible that in the first execution the tests do not go through the lack of an incorrect source code or adaptation. In this case, you should analyze if the adaptation performed is correct and feature has all the code for its execution, fixing any errors and running again the unit test execution commands until the test passes without errors, as can be seen in the figure below. See how to perform the unit tests in the next section.

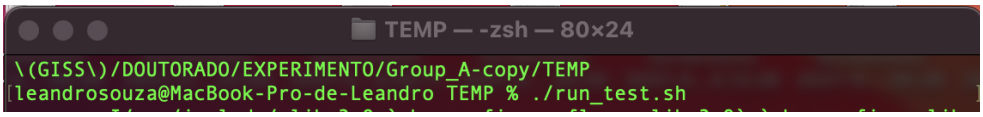
OBS: The proper feature compilation process will generate an individual file.x. Only after the generation of this file and subsequent execution of failure -free tests, as per section 6, we can say that the feature has been correctly extracted from the donor and its files can be inserted into the host system directory.

3. **Mergin:** Once the code is extracted and the unit tests are performed you can enter the feature insertion into the host environment. This process consists of inserting a call to the insertion point replacing the notation __ADDGRAFTHERE__JUSTHERE by the chamda to the *dir_init()* function. You must access the *vi.c* file, within the host system directory, and identify the notation __ADDGRAFTHERE__JUSTHERE. This annotation signals the insertion point of a call to the feature. Instead of this notation you should insert a call to the *dir_init* function, adding the parameters, if necessary, among the local variables declared in the main function.

Once all these steps are completed and feature is passing on the unit tests, you will be able to execute the host execution command and use feature in the host. The details of this process are given in Section 7 of this document.

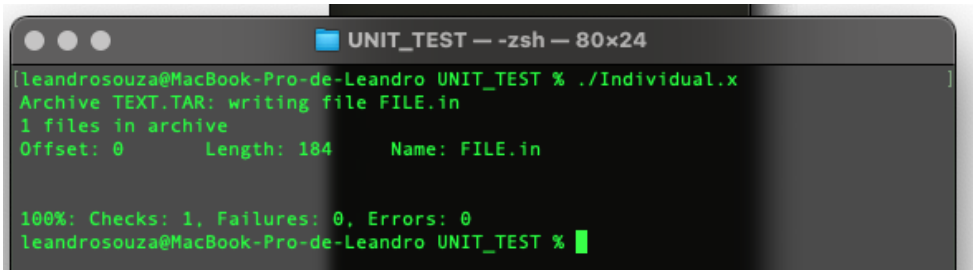
8. UNIT TEsST EXECUTION

When identifying, extracting the code elements, and performing the necessary adaptation you can perform the unit tests made available within the Test_Suite directory. For this, you must open the terminal, access the Test_Suite Directory and execute the command: `./run_test.sh` that will compile the copied feature and execute it. He may inform some Warning that you should disregard but should not generate errors. If you are informed of the compilation process, you must correct the error in the transferred code.



Terminal screen with the execution of comand ./run_test.sh.

If compilation occurs successfully and no error is informed, the above command should generate an individual file.x in the test directory. Once this file is generated it can be executed with the command: `./Individual.x` that will test the feature. If Feature is working properly, you will receive the message:



Terminal with unit testing informing: 100% percentage of acceptance; Checks: 1. Number of tests performed; Failures: 0 number of failures and errors: 0, number of errors.

9. TRANSPLANT VALIDATIOIN

After transferring the feature to the host system and it is passing on the unit tests, you can perform the final transplant validation test. We have provided a post-operative test suite. The post-operative tests correspond to a set of regression, augmented regression and acceptance tests to exercise the feature transplanted and check if the transplant has no broken the host system.

Executing the post-operative tests:

1. Open the terminal;
2. Access the directory **HOST_TO_TRANSPLANT/NEATVI_1.0/TRANSPLANTATION_TEST_CASES** where you insert the feature source code.
3. Execute the commands: `./test_product_line.sh`

10. SENDING ALL ARTIIFACTS

After the execution of the experiment is completed, make sure that it computed all the time elapsed at each stage of the process. Sign up to the researchers to end the activity and time registration worksheet.

Rename Group B Directory Complete your name and compact the folder. Then upload the folder with your changes from the form.

If you have a shipping problem, download the artifacts from transplantation_artifacts and report this to the researchers

We greatly appreciate the time and availability to perform this vital experiment for the progress of our research.