Table of Contents

Test a Stripping line with git

Foreword

If you are completely unfamiliar with git, a good place to start is the Git4LHCb web page.

When testing a stripping line, there are two possibilities. Either you want to test an existing stripping line that has been run already in a stripping campaign, or you want to develop or modify a stripping line to be run in a future stripping campaign. We consider both cases, where we assume the existing stripping line is run in S34 in DaVinci v44r4, and the line being developed is in 2018-patches.

Setup the environment

Before you can test a stripping line you need to set up the environment. For an existing stripping line, the DaVinci version is well defined with version vXXrYYpZZ. For a campaign under development, the DaVinci version is the name of the development branch, for example, 2018-patches (the other possible branches are 2017-patches, stripping21-patches, stripping21-firstpass-patches or lhcb-reco14-patches). If you are unsure, check with the Stripping liaison of your WG.

To check which platforms can be used with a particular DaVinci version, run the following command, e.g. for DaVinci v44r4:

```
lb-sdb-query listPlatforms DaVinci v44r4
```

then set the platform with:

```
lb-set-platform x86_64-centos7-gcc7-opt
```

Note that this doesn't work for campaigns under development. In this case, you can check with the Stripping liaison or the instructions for the campaign, or check the platforms that appear in the nightly builds, e.g. for 2018-patches

After you set the platform, then you are ready to set up DaVinci. If you are testing on an existing DaVinci version, you can do the following:

```
lb-dev DaVinci/vXXrYYpZZ
cd DaVinciDev_vXXrYYpZZ
```

If it's a campaign under development, then you will take the latest nightly build:

```
lb-dev --nightly lhcb-2018-patches/latest DaVinci/2018-patches
cd DaVinciDev_2018-patches
```

Checkout the relevant version of Phys/StrippingSelections

First set the project to be used, in this case "Stripping"

```
git lb-use Stripping
```

then checkout and compile StrippingSelections from the tagged branch or the currently used branch for the development of a certain Stripping campaign (check with liaisons if unsure). In this example, we assume you

are using DaVinci v44r4, in which case the Stripping tag (as shown in the DaVinci v44r4 release notes) is v12r3p1:

```
git lb-checkout Stripping/v12r3p1 Phys/StrippingSelections
make configure
make
```

If the campaign is under development, then stripping won't be tagged yet. In this case, we assume that the development branch is 2018-patches:

```
git lb-checkout Stripping/2018-patches Phys/StrippingSelections
make configure
make
```

Now you can modify the content of StrippingSelections by adding or modifying a line; every time you do it, the package must be recompiled.

```
make purge
make
```

Add a new line

If you are modifying an existing line you can skip this part. The lines in StrippingSelections are organised in WGs. Suppose we want to add the module StrippingMyCharmDecay.py to the lines of the Charm WG. We need to copy StrippingMyCharmDecay.py to the Phys/StrippingSelections/python/StrippingSelections/StrippingCharm/ directory

```
cp <path_of_your_file/StrippingMyCharmDecay.py Phys/StrippingSelections/python/StrippingSelection
```

Since this is a new line, it has to be added to `Phys/StrippingSelections/python/StrippingSelections/StrippingCharm/__init__.py` in the, open the init.py file with your favourite editor and add your line to the end of the "_selections" list at the beginning of the file (omit the .py extension)

```
_selections = [ "StrippingSomeCharm","StrippingSomeOtherCharm", "StrippingMyCharmDecay"]
```

Now you are ready to recompile StrippingSelections, so as usual do

```
make purge
make
```

If you get compilation errors or a warning due to your line please fix them and then recompile. If get an error/warning NOT due to your line, please inform the stripping coordinators.

Modify an existing line

To modify a an existing module is sufficient to apply the changes using your favourite editor and then compile again

```
make purge
make
```

Test a stripping line

To set up the correct environment variables, issue the command

```
./run bash --norc
```

This will bring you to a new bash shell with the correct environment settings (e.g. you will be able to use gaudirun.py). You will need to do this every time that you recompile the package.

The test scripts for users can be found under $STRIPPINGSELECTIONSROOT/tests/users. Among others you can find one called TestMyStrippingLine.py. To test your line by simply editing the file to set the variable 'confname' with the value of the key 'NAME' in your default_config (or directly the key identifying your specific configuration as explained here). So if the name of your configuration is "MyCharmLines" you have simply to set

```
confname='MyCharmLines'
```

At this point you can finally test your line by doing, from any location

```
gaudirun.py -T Phys/StrippingSelections/tests/users/TestMyStrippingLine.py >& someOutputDir/Strip
```

the -T option allows for a better handling of the memory of the process and it's not strictly necessary for testing purposes but it's recommended.

The output of the script will be in the log file, and contains the usual printout of StrippingReport with retention and timing of your lines. In addition, the output DST and/or micro-DST will be produced according to the settings you specified in the default_config (MDST.DST included). The TES location at which your lines are saved together with all related info and flavour tagging info (if requested) are the same as the stripping that will eventually run in production so that you can run your ntuple-maker script on these DSTs to check that all the information is stored correctly. If for some reason you are unhappy with the configuration or implementation of your lines, just go back issuing the command

```
exit
```

This will bring you to the original bash session where you can apply your modifications, recompile and start again testing as in explained in this section.

If you are returning with a "fresh" lxplus session, return to the

```
DaVinciDev_vXXrYY.pZZ
```

directory, and you can run as follows:

```
./run gaudirun.py -T Phys/StrippingSelections/tests/users/TestMyStrippingLine.py >& someOutputDir
```

IMPORTANT

Please do not import anything explicitly from the StrippingSelections package, which is used for development. In production the LineBuilders are copied into StrippingArchive to freeze them. Explicit imports from StrippingSelections break this scheme and may cause your line to crash if there are differences with what is put in the StrippingArchive and what is in StrippingSelections.

Before I commit my stripping line

Once you have successfully tested your line inform your WG group liaison about the results of the testing and ask for permission to commit to git, specifically we want to know the rate of the new line and the timing information. For MDST lines, the rate should be <0.5% and for DST lines, <0.05%. For new lines, please inform also the coordinators before committing the new code. Add the results of the tests as comment to the

merge request.

Commit a new or modified line

*You must commit to a new branch and NEVER to the master of the project. The branch will be created in the moment of pushing the changes. Immagine that the name of the line is my_new_branch (please use something that you can identify afterwards), and that I need to change a couple of python files called `file1.py` and `file2.py`.

If the files are new first add them to the repository

```
git add file1.py file2.py
```

Then, issue the following two commands

```
git commit -m "message that identifies my commit"
git lb-push Stripping my_new_branch
```

Prior to making a merge request, merge changes from the master branch into your updated branch with

```
git fetch --all
git lb-checkout Stripping/master Phys/StrippingSelections
```

If there are any conflicts, resolve them and test again. To mark a conflicting file as resolved do

```
git add filename
```

then commit and push again to the same branch

```
git commit -m "Merging updates from master & resolving conflicts"
git lb-push Stripping my_new_branch
```

Were is my commit?

In the gitlab web page https://gitlab.cern.ch/lhcb/Stripping you can find all of the existing branches under **Repository/Branches**. Once you are satisfied with the results of your work you can ask to merge your branch with the master, so the changes or new lines will be executed in the next stripping campaign. To do so, click on the box **Merge Request** that follows the name of the branch, and the coordinators will check your changes and eventually accept it. You will be contacted by the coordinators in case that they found something to be changed.

REMEMBER ALWAYS TO ASK FOR THE MERGE REQUEST ONCE YOU ARE SATISFIED WITH THE CHANGES. AND CHECK HOW THAT YOUR MERGE REQUEST CONTAINS ONLY YOUR OWN CHANGES!
