commandCompiler

commandCompiler

Parse yaml files into commands for command executor

FUNCTIONS

__init__

```
def __init__(
self,
yaml_cmds
 =
None,
yaml_projects
 =
None,
target
 =
None
)
```

Setup class

listCmds

```
def listCmds(
self
)
```

Print a list of all the commands available

listProjects

```
def listProjects(
self
)
```

Print a list of all the

clear

```
def clear(
self
)
```

Clear results of create and current yaml file pointer.

setProjects

```
def setProjects(
self,
yaml_projects
)
```

Set a yaml file for project commands

setCmds

```
def setCmds(
self,
yaml_cmds
)
```

Set a yaml file for commands available

create

```
def create(
self,
yaml_cmds
  =
None,
yaml_projects
  =
None,
target
  =
None
)
```

Pass a yaml file to use for processing into format for commandExecutor.

getResult

```
def getResult(
self
)
```

Return dict of dicts that contains lists with lists of lists to execute with subprocess {'project': { 'concurrent': [[["make", "def_config"], ["make"]], [["fusesoc", "run", "--build", "--target", "zed_blinky", "::blinky:1.0.0"]]], 'sequential': [[]]}}

_checkTarget

```
def _checkTarget(
self
)
```

_process

```
def _process(
self
)
```

Create dict of dicts that contains lists with lists of lists to execute with subprocess {'project': { 'concurrent': [[["make", "def_config"], ["make"]], [["fusesoc", "run", "--build", "--target", "zed_blinky", "::blinky:1.0.0"]]], 'sequential': [[]]}}

_load_yaml

```
def _load_yaml(
self,
yaml_file
)
```

Internal function that will load the yaml file that contains project commands or commands.

commandExecutor

commandExecutor

Execute commands create from commandCompileris not a valid selection

FUNCTIONS

clear

```
def clear(
self
)
```

Clear state of commandExecutor to init with no values passed

completed

```
def completed(
self
)
```

return a bool if all items have been completed.

failed

```
def failed(
self
)
```

return a bool indicating failure

stop

```
def stop(
self
)
```

Kill and current processes in build threads

setDryrun

```
def setDryrun(
dryrun
 =
False
)
```

Setting to True sets the project run everything up to the subprocess.Popen call, which is skipped.

setProjects

```
def setProjects(
self,
projects
)
```

Set the projects for the executor to build

runProject

```
def runProject(
self,
projects
 =
None
)
```

Call the internal execute method to start calling up each of the projects to build.

_execute

```
def _execute(
self
)
```

Call subprocess as a thread an_gen_build_cmdsd add it to a list of threads for wait to check on. iterate over projects avaiable and execute commands per project

_subprocess

```
def _subprocess(
self,
list_of_commands
```

```
)
```

Responsible for taking a list of commands and launching threads concurrently or singurely.

_project_cmd_count

```
def _project_cmd_count(
self,
run_types
)
```

Number of commands in a project.

_thread_exception

```
def _thread_exception(
self,
args
)
```

Used to kill all threads once one has failed.

_bar_thread

```
def _bar_thread(
self
)
```

Creates progress bar display in terminal for end user display.