Build and Run cFS Tutorial

Objectives

- Describe Basecamp's core Flight System (cFS) build and runtime environments
- Introduce Application Specifications and describe how they are used in the build and runtime environments
- Application Specifications include Electronic Data Sheet (EDS) interface definitions and JSON application integration parameter definitions

Basecamp supports cFS application related activities and does not address porting the cFS to different processor/operating system platforms.

- https://github.com/cfs-tools/cfs-platform-list provides a list of community supported cFS ports

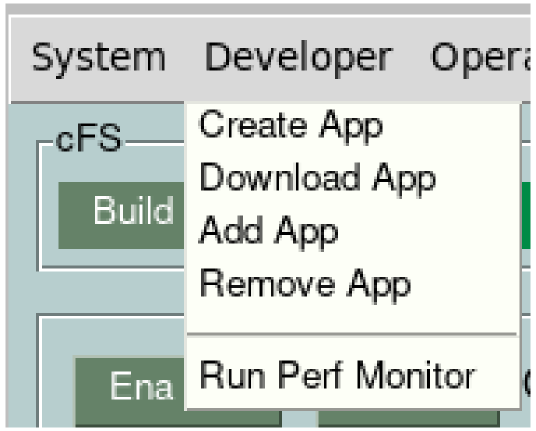
Lesson 1

Objectives

- Describe Basecamp’s build environment
- Introduce startup and runtime environment

Notes

- No manual configurations are required
  - to build and run the default Basecamp cFS target
  - to add/remove apps when using the Devloper menu



Key cFS Build Directories

- Below are some of the important top-level directories and files used when building the cFS
  - Other directories and files will be introduced as needed

```
cfs-basecamp
├─apps/  . . . . . . . . .  Preinstalled Basecamp apps that provide essential functionality
├─cfe-eds-framework/
│ ├─apps/  . . . . . . . .  cfe-eds 'lab' apps, all but Command Ingest are replaced by Basecamp apps
│ ├─build/ . . . . . . . .  Output directory containing the artifacts produced by the build process
│ ├─cfe/ . . . . . . . . .  core Flight Executive, includes Electronic Data Sheet base type definitions
│ ├─basecamp_defs/ . . . .  Top-level cmake files that define the cFS target
│ ├─libs/
│ ├─osal/
│ ├─psp/
│ └─tools/ . . . . . . . .  Electronic Data Sheet build tools
└─usr/
  └─apps/  . . . . . . . .  User apps added to Basecamp
```

Key Build Files

```
cfs-basecamp
├-apps/
├-cfe-eds-framework/
│ ├-apps/
│ ├-build/
│ │ ├- exe/cpu1/core-cpu1 . . . . . . .  Core Flight Executive (cFE) executable
│ │ └- exe/cpu1/cf/*.*  . . . . . . . .  “Compact Flash (CF)” contains app/lib object files and app default table files
│ ├-cfe/
│ ├-basecamp_defs/
│ │ ├-global_build_options.cmake   . .  Defines app source search path
│ │ ├-mission_build_custom.cmake   . .  Defines compiler options
│ │ ├-targets.cmake   . . . . . . . . .  Defines the target name cpu1, the apps/libs in the cpu1 target, and files to copy
│ │ │                                    from basecamp_defs/ to build/exe/cpu1/cf/
│ │ └-cpu1_cfe_es_startup.scr . . . .  Defines the apps/libs that are loaded when core-cpu1 runs
│ ├-libs/
│ ├-Makefile  . . . . . . . . . . . .  Controls the build process; GNU-make wrapper that calls the CMake tools
│ ├-osal/
│ ├-psp/
│ └-tools/
└-usr/
  └-apps/
```

There are multiple Electronic Data Sheet files used during the build process that are described in Lesson 2

Cmake Structure

cFS Mission

- Top-level context/scope that includes everything that is being built
- Includes development tools (not crossed compiled) that are built within this context
- Includes one or more cFS "**target**" CPUs that are built using an architectural (**"arch")** specification

basecamp_defs/targets.cmake

- Defines one or more cFS targets that is identified as cpu1, cpu2, etc.
- Each target is built using an "arch" cmake specification
- See targets.cmake file prologue comments for more information
- Basecampd defaults to cpu1 being built for a Linux platform

Custom cmake configurations in basecamp_defs directroy:

- The following files allow you to call "add_compile_options" or "add_definitions" as necessary for your use case
  - global_build_options.cmake
  - mission_build_custom.cmake
  - arch_build_custom.cmake
- The difference is scope - global applies globally, mission applies only to mission (host) build, and arch applies only to target build. The latter can also be broken down per-arch.

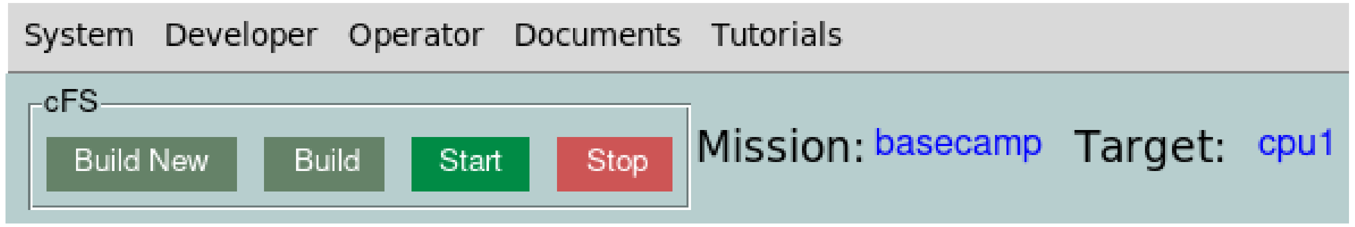
cfe-eds-framework/Makefile

- Single access point to manage build process
- Basecamp's custom "**topicids**" make target runs a tool to resolve EDS Topic IDs in app initialization JSON file (described in lesson 2)

Basecamp’s target.cmake

- **Target.cmake defines configurations for a mission’s cFS targets**
- **Basecamp defines one mission with a single target using the following commands and these names are shown on the main window**

```
SET(MISSION_NAME “Basecamp")
SET(MISSION_CPUNAMES cpu1)
```



- **The following configurations define Basecamp’s cpu1 target contents**

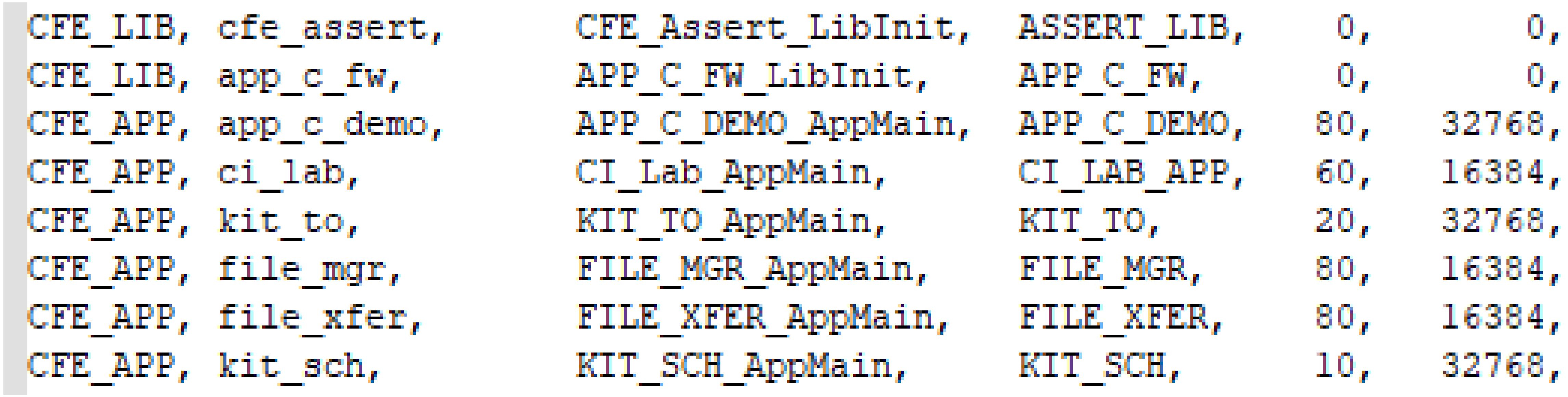
```
list(APPEND MISSION_GLOBAL_APPLIST app_c_fw)
SET(cpu1_APPLIST ci_lab kit_to kit_sch file_mgr file_xfer app_c_demo)
SET(cpu1_FILELIST cfe_es_startup.scr file_mgr_ini.json file_xfer_ini.json . . .
```

  - The term ‘app’ refers to both libraries and applications
  - Basecamp’s application framework, app_c_fw, is defined globally so it is accessible to all targets if new targets are added
  - New libraries and applications are added to *cpu1_APPLIST*
  - *cpu1_FILELIST* defines which files are copied from the basecamp_defs directory to the build/exe/cpu1/cf

cfe_es_startup.scr

- cfe_es_startup.scr defines which libraries and apps are loaded during initialization
- The filename in basecamp_defs is cpu1_cfe_es_startup.scr to associate it with target cpu1 and the cpu1_ prefix is removed when the file is copied to the build/exe/cf directory

Object Type Path/Filename Entry Symbol cFE Name Priority Stack Size



```
CFE_LIB, cfe_assert,     CFE_Assert_LibInit,   ASSERT_LIB,   0,      0,
CFE_LIB, app_c_fw,       APP_C_FW_LibInit,     APP_C_FW,     0,      0,
CFE_APP, app_c_demo,     APP_C_DEMO_AppMain,   APP_C_DEMO,  80,  32768,
CFE_APP, ci_lab,         CI_Lab_AppMain,       CI_LAB_APP,  60,  16384,
CFE_APP, kit_to,         KIT_TO_AppMain,       KIT_TO,      20,  32768,
CFE_APP, file_mgr,       FILE_MGR_AppMain,     FILE_MGR,    80,  16384,
CFE_APP, file_xfer,      FILE_XFER_AppMain,    FILE_XFER,   80,  16384,
CFE_APP, kit_sch,        KIT_SCH_AppMain,      KIT_SCH,     10,  32768,
```

- On a Linux system the core Flight Executive executes as a process and each app is a thread

Runtime Applications

- Basecamp’s **Scheduler** and **Telemetry Output** apps provide functionality that support apps to run as part of a cFS target

- The Scheduler app send messages on the Software Bus at fixed intervals to signal apps to perform a particular function

- Telemetry Output receives messages from the Software Bus and sends them to an external destination

- The Basecamp Application Developer’s Guide and each app’s documentation provide complete details on these two app’s tables

Scheduler App Tables

- **The kit scheduler (KIT_SCH) app uses a scheduler table defined in cpu1_kit_sch_schtbl.json to determine when to send messages on the software bus**

- **Apps use these messages to perform periodic functions**
  - Apps are not required to use KIT_SCH
  - An app's JSON initialization table defines which messages (EDS topic IDs) are used by the app (Details in Lesson 2)

- **Basecamp's default scheduler table divides a second into four 250ms slots and each slot can have up to 15 messages sent**

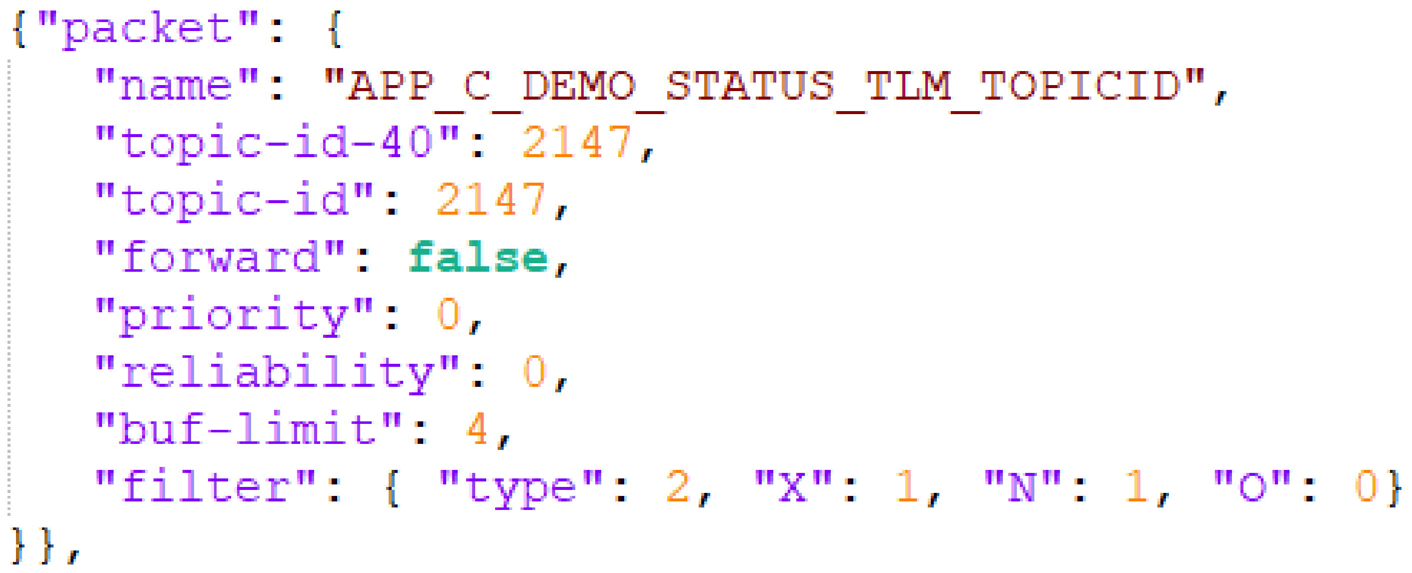
- **The following periodic messages are provided by Basecamp and available for apps to use**
  - BC_SCH_1_HZ_TOPICID
  - BC_SCH_2_HZ_TOPICID
  - BC_SCH_4_HZ_TOPICID
  - BC_SCH_2_SEC_TOPICID
  - BC_SCH_4_SEC_TOPICID
  - BC_SCH_8_SEC_TOPICID

Telemetry Output Table

- **The kit telemetry output (KIT_TO) app uses a packet filter table defined in cpu1_kit_so_pkt_tbl.json to determine which telemetry messages are read from the software bus and sent to an external systems over a UDP port**

- **Apps define telemetry message topic IDs in their EDS definitions**

- **AP_C_DEMO's status telemetry packet definition** (See Basecamp Application Developer's Guide for details)



```
{"packet": {
   "name": "APP_C_DEMO_STATUS_TLM_TOPICID",
   "topic-id-40": 2147,
   "topic-id": 2147,
   "forward": false,
   "priority": 0,
   "reliability": 0,
   "buf-limit": 4,
   "filter": { "type": 2, "X": 1, "N": 1, "O": 0}
}},
```