Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

The OPUS Pipeline Toolkits

C. Boyer, T. H. Choo

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21218

Abstract. The OPUS pipeline, which employs a blackboard architecture, has been processing Hubble Space Telescope (HST) data for nearly a year. OPUS was designed for both reusability and extensibility, as well as portability to different platforms and projects. OPUS contains a toolkit of resource files and programs which provide the users with the ability to customize their own pipeline. ASCII resource files can be used to define the configuration of the system, and to add processes to the pipeline dynamically. The OPUS callable routines provide applications with even more flexible methods for interfacing with the OPUS blackboard. This paper will discuss how the OPUS toolkit—both the resource files and the software libraries—is used to configure an OPUS data processing pipeline.

1. Introduction

The purpose of the OPUS toolkit is to provide a mechanism for processes to be easily incorporated in the OPUS data processing pipeline (Rose et al. 1994). The OPUS toolkit contains software built upon the OPUS resource file concept, which allows the user to tailor the pipeline without any software changes. The toolkit contains C callable routines that provide access to the pipeline's blackboards and the OPUS resource files, GUI applications for monitoring and managing the blackboards, and an OPUS shell that can be used to dynamically add a third party software application to the OPUS pipeline.

Currently three types of resource files—the pipeline stage file, the process resource file, and the path—file are supported. The pipeline stage file defines the processing steps, or "stages," available in the pipeline. Each processing stage is described in a process resource file. This file contains the specific behavior, and input/outputs, of a processing step. Finally the path file describes the data flow of the pipeline: directories where data are stored and obtained, and global pipeline information.

Resource files allow the user to customize the scope of the pipeline, process specific attributes, and disk management, using only a text editor. It is possible to create pipelines that feed data to other pipelines, pipelines that merge or separate based on the data being processed, and pipelines that distribute the finished data product based on some characteristic of the data. The OPUS toolkit is built on both the existence of these resource files and access to the blackboard.

The OPUS system employs two blackboards to support the communication between pipeline and operators. One blackboard is used for processes activities, and the other is used to track the progress of observations in the pipeline. The

C callable toolkit contains two packages (the Process Status Flag (PSF) package and Observation Status Flag (OSF) package) that allow an application to interface directly with these blackboards. The C callable toolkit also contains a resource package that is used to manage the OPUS resource files, needed to support the OPUS distributed pipeline.

The GUI applications that come with the toolkit are the Process Status Manager and Observation Status Manager. The Process Manager (PMG) is used to monitor the PSF blackboard and to send command messages to processes. The OMG (OSF Manager), on the other hand, is used to monitor and manage the OSF blackboards. Both the PMG and OMG are discussed in detail in Rose et al. (1995).

2. Process Status Flag (PSTAT) Package

Each process in the OPUS pipeline is assigned a PSF message on process initialization. The PSTAT package provides an interface to the PSF message on the blackboard.

The PSF message consists of the components, PID, Process name, Activity status, Start Time, Reinitialized Time, Path name, CPU, and Command. The PID is set at process initialization with the process ID assigned by the operating system. The process name is the name of the process, and it is the same as the process resource file name. The activity status is updated by the process to reflect the current activity, and it could be set to, for example, INITIALIZING, SUSPENDED, WORKING, or Observation name. The start time is the process start time. The reinitialized time is time when the process last reinitialized itself. The path name is the name of the path that the process runs on. The CPU is the name of the CPU which executes the process. The command is the current user directive. This field is set by user, and it may be set to HALT, SUSPEND, RESUME, or REINITIALIZE. The process then reacts to this command, and performs the appropriate action.

Accessing the PSF message package must be done through the PSTAT package. The PSTAT package contains functions that allow creation, modification, deletion, and searches of PSF messages on the blackboard. As multiple PSF messages are updated, created, and removed by processes and users, message contention can become an issue. The PSTAT package internally resolves the message contentions on the blackboard.

3. Observation Status Flag (OSF) Package

The Observation Status Flag (OSF) package provides an interface to the OSF blackboard. Each exposure that enters the pipeline is assigned a unique OSF which is used to track the progress of observation processing.

An OSF message consist of the components, Start time, Status, Dataset name, Data ID, DCF, and Command. The Start time is the OSF creation time. The Status shows the progress of an observation through the pipeline stages. The field can be retrieved as a whole or broken down to derived stages in the pipeline.stage. As an observation proceeds down a pipeline, different stages of the status field can be updated by various processes. The Dataset name is the name of the observation. The Data Class assigns an OSF to a class of data. The

DCF field is a HST specific number and refers to the number assigned by DCF to the exposure. The Command is the current user directive. For example, the user may set this field to "HALT," halting an exposure from further processing. Processing is restored via the "RESUME" directive.

The OSF package also contains functions that allow creation, modification, deletion, and searches of OSF message on the blackboard. Since OPUS is a distributed and parallel processing pipeline, multiple processes may update the same OSF message at the same time. The OSF package resolves the simultaneous accessing of the same OSF message, and encapsulates the message contention from the application.

4. Process Resource File (PRSC) Package

The PRSC package provides an interface to the OPUS resource files. The OPUS process resource file defines the specific behavior of a process without regard to the rest of the processes in the pipeline. All processes are run in a path defined by a path resource file. Resources in the process resource file can be superseded by the resources in the path file. This allows the user to enforce specific behaviors for all processes selected to run in the same path.

The PRSC package contains routines that perform the symbol substitution, and resource hierarchy superseding, and maintain the resources in a dynamic structure that can be easily retrieved by the application at run time. An example of symbol substitution, and resource superseding performed by the PRSC package is shown below. These are the values in path and resource files:

`RED.PATH` File	`INGEST.RESOURCE` File
`*.RETRY = 5`	`RETRY = 1`
`LEVEL = LEVEL1`	`UPDATE = LEVEL`
`INGEST.LEVEL = LEVEL5`	
	`DATA_DIR = disk$scratch:[in]`
`HRS_DIR_1 = disk$sratch:[red]`	`IN_DIR = HRS_DIR_1`

Keyword values from the PRSC package for the INGEST process are:

`RETRY = 5`	The `RETRY` value specified in the `INGEST.RESOURCE` file, is superseded by the value in `RED.PATH`
`LEVEL = LEVEL5`	The symbol `LEVEL` is substituted with value of `ST.LEVEL` specified in the `RED.PATH`
`IN_DIR = disk$scratch:[red]`	The symbol, `HRS_DIR_1` is substituted `with disk$scratch:[red]` from `RED.PATH` file.
`DATA_DIR = disk$scratch:[in]`	The `DATA_DIR` value in `INGEST.RESOURCE` is not substituted or overriden by any path file values.

5. OPUS Shell

OPUS Shell is an executable that allows third party applications that have no knowledge about the OPUS environment to run in the pipeline. The third party software can be a VMS DCL script, IRAF script, or an executable. The OPUS shell manages both the blackboards and the resource files on behalf of the third party software. A process is added to the pipeline by making an addition to the pipeline.stage file and creating a process resource file. The pipeline.stage file assigns a character of the OSF's status column to track process activities. The process resource file defines the type of OPUS event that triggers the OPUS shell into action. Currently OPUS supports an OSF event, a time event, file event, and a command event. A single process can be made to respond to multiple types and instances of OPUS events.

An OSF event is the presence of an OSF message of a certain pattern on the blackboard, and the message pattern is described in the process resource file. When the OPUS shell encounters the OSF event, it will update the PSF message Status field with the observation dataset name, and then execute the third party software. On completion of the third party software, the OPUS shell will reset the PSF's Status field to IDLE and update the OSF message to reflect the processing result.

A File Object event is defined as the presence of a file whose file name pattern and directory are defined in the process resource file. Whenever there is a File Object event, OPUS shell will first update the PSF's Status field with the name of the file object before executing the third party software. On completion, it will reset the PSF's Status field to IDLE.

OPUS shell can also be configured to execute the third party software based on a time interval, and this is called a time event. The interval between execution can be set in the process resource file.

A command event is defined as the presence of a non-blank string in the PSF's Command field. The OPUS shell has been configured to recognize the following predefined commands: SUSPEND, HALT, and RESUME. When a HALT command is detected, the OPUS Shell terminates the process. If a SUSPEND command is detected, OPUS shell will update the PSF's Status field to SUSPENDED, and the OPUS shell remains in a sleep mode until a RESUME or HALT command is detected.

References

Rose, J., Choo, T. H., Rose, M. A. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 311

Rose, J., et al. 1995, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne, & J. J. E. Hayes (San Francisco: ASP), 429