# Wide-field Infrared Survey Explorer (WISE)

# Data Processing Operations Procedures

**Version [1.0]**

**28-October-2009**

**Prepared by: Ron Beck**

**Infrared Processing and Analysis Center**
**California Institute of Technology**

**WSDC D-C005**

**Approved By:**

Roc Cutri, WISE Science Data Center Manager

Tim Conrow, WISE Science Data Center Lead Engineer

## Revision History

| Date | Version | Author | Description |
|---|---|---|---|
| 10/02/09 | 0.1 | Ron Beck | Initial Draft |
| 10/28/09 | 1.0 | Ron Beck | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## Table of Contents

# 1 INTRODUCTION

## 1.1 Document Scope

This document will explain the procedures for Data Processing Operations. Once the raw data has been ingested, we can run the scan pipeline processing on it. We expect to receive about 32 scans per day containing around 250 frames worth of raw images in four deliveries. The scan pipeline processing runs the frame pipeline on all frames and then will run a number of modules on the entire scan. Once all the scans for a region of the sky have been processed, we can run the one degree square coadds that cover that region including the source lists. When the coadds and source lists are created, we will have scheduled deliveries of the scans, coadds and lists to IRSA so the data can be made public.
We are responsible for the raw data archive which means multiple backups and sending copies offsite. We need to manage resources - disk space, cpus, condor (the pipeline distribution system) and troubleshoot network issues.

## 1.2 Applicable Documents

http://wise.ipac.caltech.edu/wiki/index.php/Nodes_command
http://wise.ipac.caltech.edu/wiki/index.php/Node_clean_command
http://wise.ipac.caltech.edu/wiki/index.php/Scan_summary_command
http://wise.ipac.caltech.edu/wiki/index.php/Compile_mframe_logs_command
http://wise.ipac.caltech.edu/wiki/index.php/Qlook_summary_command
http://wise.ipac.caltech.edu/wiki/index.php/Broken_links_command
http://wise.ipac.caltech.edu/wiki/index.php/Add2scans_command
http://wise.ipac.caltech.edu/wiki/index.php/Run_scans_command
http://wise.ipac.caltech.edu/wiki/index.php/Showme_running_command
http://www.cs.wisc.edu/condor/manual/v6.2/8_Command_Reference.html#command-reference

## 1.3 Acronyms

IPAC - Infrared Processing and Analysis Center, California Institute of Technology
MOS - Mission Operations System
WSDC - WISE Science Data Center (IPAC)
WSDS - WISE Science Data System

# 2 Scan Pipeline Procedure

## 2.1 run_scans Command

Command /home/beck/bin/run_scans is the scan distribution script to be run on wcnode01. This script will run continuously in operations. In dir /wise/fops/operations, there is a file "scan". run_scans will continuously (every 10 seconds) check this file for scans to run. The scan file can take two different forms. There can be entries with just the scan identifier or a scan identifier with the wsspipe command to be run.

Here is an example of both flavors ...

```
beck@caustic;rhe4():operations[0]% cat scan
01248a
01248a pass "wsspipe -ind . -cluster -preclean all -steps @all,+dynacal"
beck@caustic;rhe4():operations[0]%
```

Any entries with just the scan identifier will be run with the following exec.

```
wsspipe -v -cluster -preclean all -mkdir -ind . -frnums -1
```

When the run_scans script sees the pass parameter, it knows to run the attached wsspipe command verbatim instead of the vanilla exec.

The run_scans script will monitor the condor job queue using the condor_status command. When the number of jobs returned from this command dips below 400, the script will look to start another scan from the scan file. This is an attempt to keep all the node slots busy continuously.

run_scans has an optional parameter which is the max number of scans to run at once and defaults to 8.

### 2.2 run_scans Miscellaneous files

The run_scans script creates or updates numerous files. These files are located in the /wise/fops/operations dir. File exec.status keeps track of the scan start and stop times. Here is a sample.

```
beck@caustic;rhe4():operations[0]% tail exec.status
Jul 13 16:37:39 01248a start
Jul 13 16:45:37 01248a complete
Jul 16 11:35:28 01248a start
Jul 16 11:59:30 01248a complete
Jul 16 13:11:51 01248a start
Jul 16 13:42:17 01248a complete
Jul 16 15:19:24 01250a start
Jul 16 15:28:54 01250a complete
beck@caustic;rhe4():operations[0]%
```

This comes in handy if you want to know when a scan was run or how many times.

There is a gostop file where you can control whether more scans are started from the scan file. 0 means don't start any more, 1 means let em rip. This is queried every 10 seconds by run_scans.

### 2.3 Scan Summary Script

When a scan completes, the run_scans script fires off another script that goes through all the log files for the scan including all the frame pipe logs and outputs them to files in the /wise/fops/operations/problems dir. the files take the form SCAN.MM-D-HH:MM:SS.PROBS where PROBS is either ok or errors.

Below is a directory listing with scan summary files.

```
beck@caustic;rhe4():problems[0]% pwd
/wise/fops/operations/problems
beck@caustic;rhe4():problems[0]% \ls -1 | head
00433a.06-30-14:55:14.errors
00434a.06-30-15:48:44.errors
00434a.07-1-08:12:05.ok
00435a.06-30-15:49:13.errors
00435a.07-1-08:13:16.ok
00436a.06-30-15:55:21.errors
00436a.07-1-08:18:39.ok
00437a.06-30-19:46:30.ok
00438a.06-30-19:40:56.ok
00439a.06-30-19:51:41.ok
beck@caustic;rhe4():problems[0]%
```

The *.ok scans return '0' code for all pipeline modules whereas the *.errors files had some non-zero codes returned.

Below is a sample ok listing.

```
beck@caustic;rhe4():problems[1]% m 12345a.06-13-18:37:18.ok
FRAME      START          ELAP STAT  SIG CODE HOST  PROGRAM
     09/06/14_01:24:44  12:13    0    0    0   caustic WSSPipe
 001 09/06/14_01:24:57  05:18    0    0    0   10 WSFPipe
 001 09/06/14_01:35:53  00:04    0    0    0   20 WSFPipePost
 002 09/06/14_01:24:57  05:23    0    0    0   10 WSFPipe
 002 09/06/14_01:35:53  00:10    0    0    0   04 WSFPipePost
 003 09/06/14_01:24:57  10:20    0    0    0   02 WSFPipe
 003 09/06/14_01:35:54  00:09    0    0    0   11 WSFPipePost
 004 09/06/14_01:24:57  05:21    0    0    0   11 WSFPipe
 004 09/06/14_01:35:55  00:08    0    0    0   12 WSFPipePost
 005 09/06/14_01:24:57  04:15    0    0    0   20 WSFPipe
 005 09/06/14_01:35:55  00:11    0    0    0   08 WSFPipePost
 007 09/06/14_01:24:57  05:25    0    0    0   10 WSFPipe
 007 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 008 09/06/14_01:24:57  10:31    0    0    0   02 WSFPipe
 008 09/06/14_01:35:54  00:10    0    0    0   04 WSFPipePost
 009 09/06/14_01:24:57  05:19    0    0    0   11 WSFPipe
 009 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 010 09/06/14_01:24:57  04:14    0    0    0   20 WSFPipe
 010 09/06/14_01:35:55  00:10    0    0    0   08 WSFPipePost
 011 09/06/14_01:24:57  05:28    0    0    0   10 WSFPipe
 011 09/06/14_01:35:54  00:07    0    0    0   30 WSFPipePost
 012 09/06/14_01:24:57  10:34    0    0    0   02 WSFPipe
 012 09/06/14_01:35:54  00:09    0    0    0   20 WSFPipePost
     09/06/14_01:35:33  00:00    0    0    0   caustic Spawn_rotmeta
     09/06/14_01:35:33  00:13    0    0    0   caustic Spawn_wssflag
     09/06/14_01:35:33  00:13    0    0    0   caustic WSSFlag
     09/06/14_01:35:34  00:00    0    0    0   caustic Spawn_dumptbl
     09/06/14_01:35:48  00:02    0    0    0   30 WSSPCal
     09/06/14_01:35:49  00:00    0    0    0   30 Spawn_spcal
     09/06/14_01:35:51  00:17    0    0    0   caustic Spawn_wsspipepost
     09/06/14_01:35:51  00:17    0    0    0   caustic WSSPipePost
     09/06/14_01:36:19  00:27    0    0    0   10 Spawn_scansync
     09/06/14_01:36:19  00:37    0    0    0   10 ScanQA
     09/06/14_01:36:46  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:47  00:03    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:51  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:52  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:53  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:54  00:01    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:55  00:00    0    0    0   10 Spawn_rotmeta
     09/06/14_01:36:55  00:01    0    0    0   10 Spawn_dumptbl
```

The summary is sorted by the module start time and lists the return codes and elapsed wall clock time by module. An error file will have non-zero return code to be followed by the offending error messages.

## 2.4 run_scans Summary Output

The window where run_scans is running updates every 10 seconds and monitors the running scans by sifting through the WSSPipe.log file. The display shows the running scans, the currently running module and the scan elapsed time. If the scan is in the WSFPIPE step, the run_scans script breaks that down by total frames, running, complete and failed frames. Below is a sample of the screen.

```
                             scan    elapsed   step
 of    251 frames total      01248a  00:21:10  SFPIPE          Jul 17 14:40:38
       217 frames running
        33 frames finished ok
>>>      1 frames failed

sleeping 10 ...
```

## 2.5 showme_running Script

Additionally, run_scans outputs a exec.update file that looks similar to the above screen sample every 10 seconds. This works in conjunction with the showme_running script that anyone can run and see the scans progress just like the ops people.

Here is a sample output from showme_running ...

```
                             scan    elapsed   step
 of    251 frames total      01248a  00:22:14  SFPIPE          Jul 17 14:41:42
         7 frames running
       243 frames finished ok
>>>      1 frames failed
```

## 3 Multi Scan Pipeline Procedure

Once the scan frame pipelines complete, we can run the multi scan pipeline. This pipeline will run on an entire scan and applies photometric calibration and artifact identification to the lists and images.

### 3.1 Multi Scan Pipeline

Below is a sample command line for running the multi scan pipeline on scan 01707a.

```
 wmspipe -v -cluster -dataroot /wise/tops -scans 01707a \
-out_base /wise/tops/scans/7a/01707a
```

The wmspipe command submits a number of processes to the nodes that run pretty quickly on a dry cluster 2 - 3 minutes per scan.

## 4 Coadd Pipeline Procedure

Once the scans and the multiscan pipeline have been run and QA have blessed the scans, we can now create the 1.5 degree coadds. The procedure consists of the wmcpipe command which given scan and position criteria will create and execute the wmfpipe or the multi frame pipeline.

### 4.1 wmcpipe Command

Below is the wmcpipe command used to create and run the coadds for the 30 orbit simulation scans 01707a-01765a.

```
 wmcpipe -v -tail tops_300off -outb tops_300off -workloc local \
 -data_root /wise/tops -preclean all -copts cmdfile=1 \
 -elonspec date=20100113T031534,20100115T014203,elonoff=-90 -getfix \
 -:getfix '-f scan=01707a-01765a,pstat=0'
```

In this example, the criteria given is the scan range which is all the scans for the 30 orbit sim, the date that the deliveries cover and the "elonoff=-90" means only do one side of the sky. This command generated 122 wmfpipe commands submitted to the cluster. This is however only one side of the sky. This same command was run again for the other side of the sky using the "elonoff=90" parm.

The wmcpipe command tracks the progress of the wmfpipes, logging the return codes in the *-coadds.tbl in the working directory that the command is run.

## 5 IRSA Delivery Procedure

Twice weekly we will get a request for IRSA database deliveries. This entails the source lists generated by the frame pipelines and their images. We will also deliver the coadded images and their source lists. This is accomplished with the dbprep and imgprep routines.

### 5.1 DBPREP Script

The dbprep output data currently goes to /wise/data/irsa_in directory. Since the deliveries will cover a number of scans, we will use the wsdc_submit command to submit them to the cluster to run. Basically, the command you want to run just follows verbatim the wsdc_submit command. The following command was run to create the IRSA delivery for scan 00433a.

```
 wsdc_submit dbprep -frame_dirs "/wise/fops/scans/3a/00433a/fr/*" -outdir \
 /wise/data/irsa_in >& /wise/data/irsa_in/dbprep.out.00433a &
```

The -frame_dirs parm points to the frame dirs to be preped. Entering command "dbprep -help" will display all the available parms. Below are the resulting created files.

```
 beck@caustic;rhe4(dev):irsa_in[0]%  pwd
 /wise/data/irsa_in
 beck@caustic;rhe4(dev):irsa_in[0]% ls -al *00433a*
 -rw-rw-r--+ 1 beck  wise    121654 Aug  7 14:44 dbprep.out.00433a
 -rw-rw-r--+ 1 beck  wise     12733 Aug  7 14:44 wise_i1ba_frm_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise    455405 Aug  7 14:44 wise_i1ba_mch_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise     10663 Aug  7 14:43 wise_i1bc_frm_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise      1577 Aug  7 14:43 wise_i1bc_mch_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise       157 Aug  7 14:43 wise_i1bc_scn_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise    401500 Aug  7 14:43 wise_i1bs_frm_3.bartbl.00433a
 -rw-rw-r--+ 1 beck  wise 991839862 Aug  7 14:43 wise_i1bs_psd_3.bartbl.00433a
 beck@caustic;rhe4(dev):irsa_in[0]%
```

Seven files are output from dbprep. The *s_psd* is the point source data for the scan. *s_frm* is per frame metadata whereas *c_scn* is the calibration scan level metadata. The *a_frm* and *a_mch* files are the asteroids and their associations. The *c_frm* is calibration frame level metadata and *c_mch* are calibration matches.

The dbprep output files should be checked for return codes after running.

```
 beck@caustic;rhe4(dev):irsa_in[0]% grep "End of DBPrep" dbprep.out.00433a
 << End of DBPrep >> status=0
 beck@caustic;rhe4(dev):irsa_in[0]%
```

Problems need to be addressed with Tracey Evans. Sherry Wheelock is Tracey's backup.

When running a number of scans which will be the case in operations, we need to concatenate all output files except the *s_psd* file by file type. For instance, all *c_scn* files can be put together into one as below.

```
 beck@caustic;rhe4(dev):irsa_in[0]% cat wise_i1bc_scn_3.bartbl.004* >
wise_i1bc_scn_3.bartbl
```

loading 1 file instead of 60 makes life easier on the IRSA people when they load the database. The *s_psd* files are pretty big and should be packaged up by 20 scans per file.

## 5.2 DBPREP Script for Coadds

The dbprep command for coadds is similar to the frames dbprep with a couple of parameter tweaks. Below is a sample run for a coadd created out of the 30 orbit sim.

```
 wsdc_submit dbprep /wise/fops/coadds/08/0833/m654/z_ort3_day1_1/0833m654_z_ort3_day1_1 \
 -data_level 3o -outdir /wise/data/beck/irsa_in >& \
 /wise/data/beck/irsa_in/dbprep.0833m654_z_ort3_day1_1.out &
```

The "-data_level 3o" alerts dbprep that this is coadd data. Below are the resulting files created.

```
 beck@wcnode01;rhe4(dev):l3[0]% ls -al *0833m654_z_ort3_day1_1*
 -rw-rw-r--  1 beck wise     8953 Oct 26 08:49 dbprep.0833m654_z_ort3_day1_1.out
 -rw-rw-r--  1 beck wise     1960 Oct 26 08:49
wise_i3os_cdd_3.bartbl.0833m654_z_ort3_day1_1
 -rw-rw-r--  1 beck wise 22390966 Oct 26 08:49
wise_i3os_psd_3.bartbl.0833m654_z_ort3_day1_1
 beck@wcnode01;rhe4(dev):l3[0]%
```

The wise_i3os_psd_3.bartbl* file contains the pipe-delimited source data for the coadd. The wise_i3os_cdd_3.bartbl* file contains the coadd metadata. As with the frames dbprep, the coadd files should be concatenated also to ease the load on the IRSA people.

## 5.3 IMGPREP Script

The imgprep command runs basically the same as dbprep only imgprep gleans information from the images created by scan pipelines. It creates a pipe-delimited file containing position info and filenames for loading into the IRSA image server database. Again we submit the imgprep job to the cluster using the wsdc_submit command.

```
 wsdc_submit imgprep -frame_dirs "/wise/fops/scans/3a/00433a/fr/*" -outdir \
 /wise/data/irsa_in > & /wise/data/irsa_in/imgprep.out.00433a &
```

The -frame_dirs parm points to the frame dirs to be preped. Entering command "imgprep -help" will display all the available parms. Below are the resulting created files.

```
 beck@caustic;rhe4(dev):irsa_in[0]% pwd
 /wise/data/irsa_in
 beck@caustic;rhe4(dev):irsa_in[0]% ls -al | grep Sep
 drwxrwxr-x   2 beck  wise       1759 Sep  2 09:31 .
 -rw-rw-r--+  1 beck  wise       4682 Sep  2 09:24 imgprep.out.00433a
 -rw-rw-r--+  1 beck  wise     723385 Sep  2 09:24 wise_i1bm_frm_3.bartbl.00433a
 beck@caustic;rhe4(dev):irsa_in[0]%
```

imgprep creates only the one pipe-delimited file describing the images for the scan.

The imgprep output files should be checked for return codes after running.

```
 beck@caustic;rhe4(dev):irsa_in[0]% grep "End of ImgPrep" imgprep.out.00433a
 << End of ImgPrep >> status=0
 beck@caustic;rhe4(dev):irsa_in[0]%
```

Problems need to be addressed with Tracey Evans. Sherry Wheelock is Tracey's backup.

When running a number of scans which will be the case in operations, we need to concatenate all output files into one for delivery to IRSA.

```
 beck@caustic;rhe4(dev):irsa_in[0]% cat wise_i1bm_frm_3.bartbl.004* >
wise_i1bm_frm_3.bartbl
 beck@caustic;rhe4(dev):irsa_in[1]%
```

### 5.4 IMGPREP Script for Coadds

The imgprep command for coadds is similar to the frames imgprep with a couple of parameter tweaks. Below is a sample run for a coadd created out of the 30 orbit sim.

```
 wsdc_submit imgprep /wise/fops/coadds/08/0833/m654/z_ort3_day1_1/0833m654_z_ort3_day1_1
\
 -data_level 3o -outdir /wise/data/beck/irsa_in >& \
 /wise/data/beck/irsa_in/imgprep.0833m654_z_ort3_day1_1.out &
```

The "-data_level 3o" alerts imgprep that this is coadd data. Below are the resulting files created.

```
 beck@wcnode01;rhe4(dev):l3[0]% ls -al *0833m654_z_ort3_day1_1*
 -rw-rw-r--  1 beck wise     2592 Oct 26 10:07 imgprep.0833m654_z_ort3_day1_1.out
 -rw-rw-r--  1 beck wise     2356 Oct 26 10:07
wise_i3om_cdd_3.bartbl.0833m654_z_ort3_day1_1
 beck@wcnode01;rhe4(dev):l3[0]%
```

The wise_i3om_cdd_3.bartbl* file contains the coadds metadata. As with the frames imgprep, the coadd files should be concatenated also to ease the load on the IRSA people.

## 6 WISE Raw Data Backup Procedure

We are the sole repository for the WISE raw data until we ship off a copy to NSSDC. We need to backup to tape the raw MOS and HRP data we receive daily.

### 6.1 raw_backup Script

Log on to machine nyx using the wisesw account. Execute command /home/wisesw/bin/raw_backup.

```
 > ./raw_backup
 command usage: raw_backup YYDOY DRIVE TAPE

                where YYDOY is the 2 digit year and julian doy dir to backup
                      DRIVE is 0 or 1 for the dat drives on nyx to use
                      TAPE  is the outer tape lable for the directory
```

The raw_backup script will backup the raw data located in /wise/fops/ingest/delivs/YYDOY. All subdirectories in this dir will be backed up one per tape file. The DRIVE parm expects either 0 or 1 for the tape drives located on the nyx machine. This machine is used because the ingest/delivs dir is local to nyx. The directory containing the tape names, files and contents is in /home/wisesw/raw_data_tape_log. There is an entry for each file on all tapes and the columns are tape name, file number, dir backed up, size, tape drive name and write date.

Once we get a successful backup of the YYDOY dir, we need to make 2 copies of the tapes, one for a local copy, and one to be sent offsite to long term storage. The latter to be arranged by the IPAC library people. This can be accomplished using a dd command from the nyx drive 0 to 1.

## 7 Pipeline Machines Maintenance

### 7.1 nodes Script

The nodes command is handy when you want to run the same command on all node machines. The IFILE parm is optional and would contain a list of machines to run on. If no IFILE parm is supplied, file /wise/fops/operations/nodes will be used. You have to create a public key prior to running this command unless you want to be prompted for a password for each machine. Instructions below.

```
beck@caustic;rhe4(ops):~[0]% nodes

 command usage: nodes COMMANDS BACKGROUND IFILE

         where  COMMANDS is list of commands separated by ';' to
                run on all pipeline nodes
                BACKGROUND is y or n to put the command in background
                IFILE is file containing list of machines to run
                commands on (optional and defaults to wcnode01-32)
```

Here is an example.

```
beck@caustic;rhe4(ops):~[1]% nodes "uptime" n
 ssh wcnode01 uptime ...
 09:13:58 up 20 days, 18:03,  1 user,  load average: 0.28, 0.21, 0.12
 ssh wcnode02 uptime ...
 09:13:58 up 20 days, 17:57,  1 user,  load average: 0.00, 0.01, 0.00
 ssh wcnode03 uptime ...
 09:13:58 up 20 days, 17:57,  0 users,  load average: 0.03, 0.02, 0.00
 ssh wcnode04 uptime ...
 09:13:59 up 20 days, 17:58,  0 users,  load average: 0.00, 0.02, 0.00
 ssh wcnode05 uptime ...
 09:13:59 up 20 days, 17:58,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode06 uptime ...
 09:14:00 up 20 days, 17:58,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode07 uptime ...
 09:14:01 up 20 days, 17:58,  0 users,  load average: 0.01, 0.01, 0.00
 ssh wcnode08 uptime ...
 09:14:01 up 20 days, 17:58,  0 users,  load average: 0.01, 0.01, 0.00
 ssh wcnode09 uptime ...
 09:14:02 up 20 days, 17:57,  0 users,  load average: 0.00, 0.00, 0.00
 ssh wcnode10 uptime ...
 09:14:03 up 20 days, 17:58,  1 user,  load average: 0.72, 0.88, 0.81
beck@caustic;rhe4(ops):~[0]%
```

### 7.2 Creating a Public Key

Instructions for creating a public key.

```
sirius:.ssh tungn$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tungn/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/tungn/.ssh/id_rsa.
Your public key has been saved in /Users/tungn/.ssh/id_rsa.pub.
```

copy the id_rsa.pub to the remote machine $HOME/.ssh/authorized_keys

In this case, the remote machine would be caustic.

### 7.3 Node Machines Disk Usage

The scan pipelines create work directories on the node machines in the /local/wise directories fops, rtb and tops. These work directories contain intermediate pipeline processed files which would be useful for debug purposes and are linked to the /wise/[fops|rtb|tops]/scans dirs. There are two scripts to help us maintain enough space in these directories to continuously keep scan pipelines running.

#### 7.3.1 broken_links Script

Command broken_links is to be run on the wcnode machines and will check the dirs on the wcnode machine versus the work link name in /wise/[fops|rtb|tops/scans/*/*/fr/*. Output displays discrepancies with the option of deleting the broken link dirs thereby saving the node /local space.

```
beck@caustic;rhe4(ops):~[0]% /home/beck/bin/broken_links
 command usage: broken_links DELETE
         where  DELETE is y or n to delete or just display
beck@caustic;rhe4(ops):~[0]%
```

Below is an example run on wcnode10.

```
/home/beck/bin/broken_links n
```

Creates following output.

```
/compute/wcnode13/wise/fops/scans/5a/00435a/fr/051/work ...
/compute/wcnode21/wise/fops/scans/5a/00435a/fr/100/work ...
/compute/wcnode02/wise/fops/scans/5a/00435a/fr/002/work ...
/compute/wcnode11/wise/fops/scans/5a/00435a/fr/020/work ...
/compute/wcnode26/wise/fops/scans/5a/00445a/fr/144/work ...
/compute/wcnode05/wise/fops/scans/5a/00445a/fr/117/work ...
/compute/wcnode16/wise/fops/scans/5a/00445a/fr/059/work ...
/compute/wcnode14/wise/fops/scans/5a/00445a/fr/237/work ...
/local/wise/fops/scans/5a/00445a/fr/090/work link broken - dir does not exist ...
/compute/wcnode11/wise/fops/scans/2a/00442a/fr/159/work ...
/local/wise/fops/scans/2a/00442a/fr/147/work link broken - dir does not exist ...
```

All the lines that start with /compute are the link names that are in the /wise/[fops|rtb|tops]/scans/*/*/fr/* dirs or where the frame work dir actually is for that frame. The "link broken" lines mean that there is a dir on wcnode10 with no corresponding link in /wise/[fops|rtb|tops]/scans at all and therefore orphaned and a prime candidate for deletion.

#### 7.3.2 node_clean Script

Command node_clean runs continuously on the wcnode machines 2 - 32 making sure we do not run out of pipeline work space on the nodes. Log files are located in /wise/fops/operations/node_clean dir.

```
command usage: node_clean DISKPER CLEANPER SLEEP

        where  DISKPER is minimum percent used on /local to start deletes
               CLEANPER is percentage to clean down to
               SLEEP is minutes before checking
```

notes: this command will awaken every SLEEP minutes and check the /local disk used percentage versus DISKPER parm. if percent used is greater than DISKPER, all /local/wise/[fops|rtb|tops]/scans work dir names are gathered by modified date. the oldest work dirs are deleted until /local used percentage is below CLEANPER and then back to sleep. the pointer to the /local node dir in /wise/fops/scans is also deleted first checking that it indeed is pointing to the /local work directory.

node_clean also cleans up all condor log files older than 3 days that are located in the /local directory.

### 7.4 Node Machines Messages Files

The wcnode machines messages files are located in /var/log. The messages files are rotated weekly with the messages file being current and messages.1, messages.2 and so on being older versions. The messages file can be read with the "dmesg" command as in the example below.

```
beck@wcnode10;rhe4():log[0]% dmesg messages | tail -15
microcode: No new microdata for cpu 6
microcode: No new microdata for cpu 7
IA-32 Microcode Update Driver v1.14 unregistered
Linux Kernel Card Services
  options:  [pci] [cardbus] [pm]
ip_tables: (C) 2000-2002 Netfilter core team
ip_tables: (C) 2000-2002 Netfilter core team
MSI INIT SUCCESS
bnx2: eth0: using MSI
bnx2: eth0 NIC Copper Link is Up, 1000 Mbps full duplex
ip_tables: (C) 2000-2002 Netfilter core team
i2c /dev entries driver
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
flatcal[28746]: segfault at 0000000000771958 rip 000000000042f200 rsp 0000007fbfff9468
error 4
Losing some ticks... checking if CPU frequency changed.
beck@wcnode10;rhe4():log[0]%
```

This can clue us in should the node machine start having problems such as a disk going bad, disk out of space, memory problems etc.

## 8 Condor commands

Condor is the software package that we use for distributing background jobs to the node machines. The condor master runs on wcnode01 while the machines wcnode02 - 32 do the actual processing as jobs are farmed out to them by the master. The condor master and clients start up automatically when rebooted and otherwise run continuously.

### 8.1 condor_status Command

The condor_status command displays all the nodes status and a summary of claimed and unclaimed nodes. See below.

```
 beck@caustic;rhe4(ops):~[0]% condor_status

 Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

 slot1@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+08:57:43
 slot2@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+06:38:56
 slot3@wcnode02.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:07
 slot4@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+09:50:28
 slot5@wcnode02.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:09
 slot6@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.080  4020  0+01:14:24
 slot7@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+14:59:32
 slot8@wcnode02.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+13:18:06
 slot1@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+10:37:16
 slot2@wcnode03.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:56
 slot3@wcnode03.ipa LINUX      X86_64 Claimed   Busy     0.000  4020  0+10:46:57
 slot4@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+14:12:37
 slot5@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:03:53
 slot6@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:02:43
 slot7@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.000  4020  0+15:04:30
 slot8@wcnode03.ipa LINUX      X86_64 Unclaimed Idle     0.040  4020  0+00:55:03
 slot1@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+07:31:21
 slot2@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:22
 slot3@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:58
 slot4@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:06:30
 slot5@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:26
 slot6@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:14:32
 slot7@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:38:54
 slot8@wcnode04.ipa LINUX      X86_64 Unclaimed Idle     0.030  1501  0+00:55:03
 slot1@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:45
 slot2@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:05:51
 slot3@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:14:19
 slot4@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:13:17
 slot5@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:39:04
 slot6@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:09:42
 slot7@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:08:08
 slot8@wcnode05.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:04
 slot1@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:16:16
 slot2@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:47
 slot3@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:36:55
 slot4@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:46:15
 slot5@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:13:15
 slot6@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:08:05
 slot7@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:10
 slot8@wcnode06.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:05:23
 slot1@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:38:24
 slot2@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:45:59
 slot3@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+17:09:57
 slot4@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:04:57
 slot5@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:05:04
 slot6@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+19:11:14
 slot7@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+20:29:32
 slot8@wcnode07.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+05:16:48
 slot2@wcnode08.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:41
 slot3@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+11:44:04
 slot4@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:39
 slot5@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:01:14
 slot6@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:03
 slot7@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:17:21
 slot8@wcnode08.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:03:48
 slot2@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+08:21:49
 slot3@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:06:47
 slot4@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:04:23
 slot5@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:19:25
 slot6@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:15:07
 slot7@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+15:39:58
 slot8@wcnode09.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:55:03
 slot1@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.060  1501  0+10:45:09
 slot2@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:10
 slot3@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+00:32:01
 slot4@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:16
 slot5@wcnode10.ipa LINUX      X86_64 Claimed   Busy     0.000  1501  0+10:45:17
 slot6@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+02:35:22
 slot7@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.000  1501  0+05:42:25
 slot8@wcnode10.ipa LINUX      X86_64 Unclaimed Idle     0.700  1501  0+00:55:03

                    Total Owner Claimed Unclaimed Matched Preempting Backfill

        X86_64/LINUX   248     0      23       225       0          0        0

               Total   248     0      23       225       0          0        0
 beck@caustic;rhe4(ops):~[0]%
```

Another variation of the command is adding the -submitters parm. This will display the jobs submitted by user and their status. Adding the -total parm will display just the totals by user.

## 8.2 condor_q Command

The condor_q command displays the actual jobs status and owners.

```
beck@caustic;rhe4(ops):~[0]% condor_q -g


-- Schedd: wcnode11.ipac.caltech.edu : <134.4.142.213:32774>
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  54.0   bauer           8/4  23:17   0+10:52:04 R  0   1220.7 procGlob
  54.1   bauer           8/4  23:17   0+10:52:04 R  0   732.4 procGlob
  54.4   bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.6   bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.9   bauer           8/4  23:17   0+10:52:07 R  0   1953.1 procGlob
  54.10  bauer           8/4  23:17   0+10:52:07 R  0   976.6 procGlob
  54.13  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.14  bauer           8/4  23:17   0+10:52:07 R  0   976.6 procGlob
  54.15  bauer           8/4  23:17   0+10:52:07 R  0   2685.5 procGlob
  54.17  bauer           8/4  23:17   0+10:52:07 R  0   317.4 procGlob
  54.20  bauer           8/4  23:17   0+10:52:07 R  0   244.1 procGlob
  54.21  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.23  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.24  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.25  bauer           8/4  23:17   0+10:52:07 R  0   732.4 procGlob
  54.26  bauer           8/4  23:17   0+10:52:07 R  0   1220.7 procGlob
  54.27  bauer           8/4  23:17   0+10:52:07 R  0   7324.2 procGlob
  54.29  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
  54.30  bauer           8/4  23:17   0+10:52:07 R  0   1464.8 procGlob
  54.31  bauer           8/4  23:17   0+10:52:07 R  0   244.1 procGlob
  54.32  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
  54.33  bauer           8/4  23:17   0+10:52:07 R  0   1709.0 procGlob
22 jobs; 0 idle, 22 running, 0 held

-- Schedd: caustic.ipac.caltech.edu : <134.4.141.175:32785>
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
84506.0   bauer           8/4  17:40   0+16:28:14 R  0   122.1 hashGlobs -i 30orb

1 jobs; 0 idle, 1 running, 0 held
beck@caustic;rhe4(ops):~[0]%
```

## 8.3 condor_on Command

From the condor master machine wcnode01, you can run command condor_on NODE where NODE is the wcnode machine name you want to start running condor on. a condor_status command will confirm that the node slots for the machine have been started.

## 8.4 condor_off Command

From the condor master machine wcnode01, you can run command condor_off NODE where NODE is the wcnode machine name you want to stop running condor on. a condor_status command will confirm that the node slots for the machine have been stopped. Should there be processes running on the node slots when the command is issued, they will be allowed to finish before shutting down.

## 8.5 condor_rm Command

From the condor master machine wcnode01, you can run command condor_rm USER where USER is the user name associated with the processes that you want to kill. This will kill all running processes instantly and remove any jobs that have yet to start. Jobs can also be deleted by cluster number and cluster.proc. These are numbers assigned by condor when you submit your job. A condor_q or condor_status -submitters will confirm that the jobs are gone.

Condor is a very sophisticated software package with lots of bells and whistles that we will not need. We will be using it to basically control running of the batch pipelines. The following link explains all of the available commands.

http://www.cs.wisc.edu/condor/manual/v6.2/8_Command_Reference.html#command-reference