Improvements to Configuration

John (TJ) Knoeller
Condor Week 2014

Usability Goals for 8.1/8.2

- Backward compatible
- Small
- Less surprising
- More powerful

Backward compatible

- Old configuration files work the same *

  * (probably… see ‘less surprising’)
- Share configuration across versions
  - New syntax *looks* valid to HTCondor 8.0
- Detect attempts to use new syntax with 8.0
  - Optional variants of new syntax that produce errors in HTCondor 8.0 and earlier

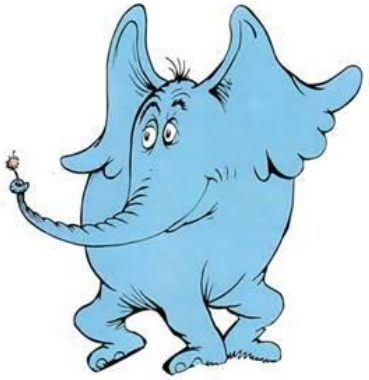
The incompatibilities

- Colon now has special meaning

```
START = true
START : true
```

  - 8.0 treats these two lines the same
  - 8.1,8.2 produces a warning for the second line
  - 8.3,8.4 will treat the second line as an error
- Interactions between comments and line continuation have changed (more later)

Small



- Smaller configuration files
  - Only deviations from defaults
- Smaller memory footprint
  - Less to parse, less to store in memory
  - Shadow more than 100 Kb smaller
- Default condor_config is 40x smaller
- Most defaults set at compile time

Bonus effects of *Small*

Compile time defaults give us:

- Shareable defaults (in the code segment)
  - Nothing to parse at startup
  - Shared between processes
- Can be easily customized downstream
  - Just edit param_info.in before you build

Less Surprising

- Defaults have been modernized
- condor_config_val sees all
- Many substitution bugs fixed
  - Couldn't build off of compile time values
  - Daemon overrides would only work sometimes
- More intuitive comment (#) and line continuation (\) interactions

Comment after line continuation

```
ALLOW_WRITE = a.b.c.d \
a.b.c.e \
# a.b.c.x \
a.b.c.z
```

- In 8.0 you end up with `#` as a list member
- In 8.1 `a.b.c.x` is commented out.

Line continuation after comment

```
# We want to frob the bobulator \
FROB_BOBULATOR = true
```

- In 8.0 \ at the end of a comment line ‘eats’ the next line, so `FROB_BOBULATOR` is not set
- In 8.1 \ at the end of a comment line is ignored, so every comment line needs its own #

Improved condor_config_val

- Shows *all* knob values, including defaults
- Remote dump works !
- So does remote verbose dump !
  - -verbose shows what file set the knob
- Dump only knobs that match a regex
- Write an ‘upgrade’ file containing *only* the knobs that you’ve changed
  - -writeconfig:upgrade <file>

More power

- New configuration language constructs
  - **`$(<param>:<default>)`**
  - **`include`**
  - **`use`** (aka meta-knobs)
  - **`if, else, elif, endif`**
- Have “backward parseable” flavors
  - **`use, include, :if`**
- Have “backward fail” flavors
  - **`@use, @include, if`**

Substitution defaults

`$(<param>:<default>)`

- Is the value of **`<param>`** if it is defined, otherwise it is **`<default>`**

  example:

  `NUM_SLOTS = $(NUM_CPUS:2)/2`

  Number of slots will be either half the number of cpus or it will be 1.

- Works in submit files…

Include

- Like LOCAL_CONFIG_FILE except
  - As many as you want
  - Nested
  - Read and parsed inline
- Can include the output of a command
- Macros on the include line substitute the current value, not the final one.

Example of Include

```
FILE = config.$(FULL_HOSTNAME)
Include : $(LOCAL_DIR)/$(FILE)
FILE = script.$(IP_ADDRESS)
Include : $(RELEASE_DIR)/$(FILE) |
Foo = bar
```

- HTCondor 8.1 Includes a file and the output of a script before parsing `Foo = bar`
- HTCondor 8.0 sees

```
FILE = script.$(IP_ADDRESS)
Include = $(RELEASE_DIR)/$(FILE) |
Foo = bar
```

Use (meta-knobs)

```
use ROLE : Submit, Execute
use POLICY : Always_Run_Jobs
use SECURITY : User_Based
use SECURITY : Strong
```

- Each keyword after colon expands inline to one or more configuration statements.
- Defined when HTCondor is built
  - See param_info.in (mentioned earlier)

Explore the meta-knobs

- Categories are currently

  `ROLE, FEATURE, POLICY, SECURITY`

- Find out what options are available with

  `condor_config_val use <category>`

- Examine contents of a meta-knob with

  `condor_config_val use <category>:<option>`

Example of Use

`use role:personal`

- HTCondor 8.1 looks up **`role:personal`**. It finds and parses this text:

```
COLLECTOR_HOST=$(CONDOR_HOST):0
DAEMON_LIST=MASTER COLLECTOR NEGOTIATOR STARTD SCHEDD
RunBenchmarks=0
```

- HTCondor 8.0 parses this as

```
use = personal
```

What did 8.0 do to role?

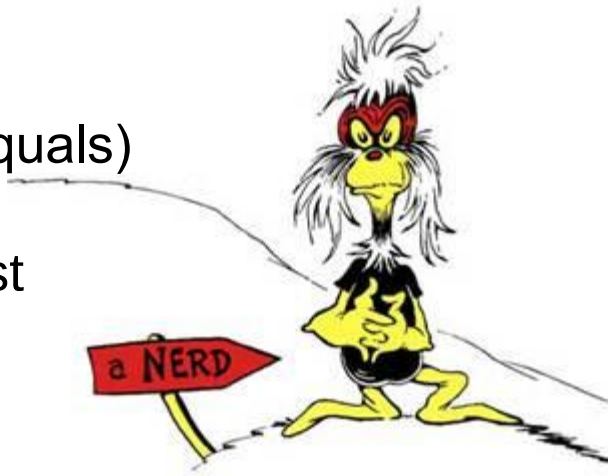
How does 8.0 turn **`use role:personal`**

into **`use = personal ?`**

From start of line, the first space, tab, colon or equal sign is the end of the knob name - “use”

From there, the first colon or equal sign is the operator “:” (which behaves just like equals)

From there, skip spaces and tabs. The rest is the value - “personal”



If / Else

- **`If, Elif`** support only basic conditionals
  - **`[!] <boolean-or-number>`**
  - **`[!] defined <name>`**
  - **`[!] version [><=]= x.y[.z]`**
- No comparison or complex conditionals
  - **`If version`** is a special case
- Conditional **`$(knob:0)`** is false when knob is not defined.

Example of If / Else

```
If version >= 8.1.6
   use feature : gpus
else
   MACHINE_RESOURCE_GPUS = 0
endif
```

- HTCondor 8.0 reports a syntax error!
  - **`else`** and **`endif`** lines have no operator

Example of If / Else for 8.0

```
:If version >= 8.1.6
:  use feature : gpus
:else
  MACHINE_RESOURCE_GPUS = 0
:endif
```

- HTCondor 8.0 only sees

  `MACHINE_RESOURCE_GPUS = 0`

  (because 8.0 ignores everything after the colon)*

CENTER FOR
HIGH THROUGHPUT
COMPUTING

HTCondor

Any Questions?