xargs

Seth House <seth@eseth.com>

Ogden Area Linux User Group

2011-02-22

srs? A whole presentation just on xargs?

That's right.

Baby steps

If not given a command, `xargs` assumes `echo`:

```
ls *.foo | xargs
```

Output the command to be run:

```
ls *.foo | xargs -t
```

Prompt before running the command:

```
ls *.foo | xargs -p
```

Why is that useful?

Get a list of process IDs:

```
pgrep ssh
```

Get info on a list of process IDs:

```
pgrep ssh | xargs ps
```

Handle arguments in groups

All at once:

```
echo 1 2 3 4 | xargs
```

One at a time:

```
echo 1 2 3 4 | xargs -n1
```

In groups of two:

```
echo 1 2 3 4 | xargs -n2
```

Call programs that take multiple arguments

```
ls *.foo | xargs -n1 -I@ cp @ ~/mybackups
```

On-the-fly shell scripts (that take arguments!)

Use `xargs` to invoke `sh -c` and have access to positional parameters:

```
ls *.foo | xargs -n1 sh -c 'echo $1' -
```

- Pass stdin as an argument to the script. This gives access to shell-script positional parameters like `$@` and `$*`.

`-n1` Handle input one-at-a-time.

Get the load for several machines

```
for ip in 192.168.0.{1..5}; \
    do ssh $ip uptime; done
```

Get the load for several machines in parallel

Get the load for several machines:

```
echo -n 192.168.0.{1..5} \
    | xargs -P5 -n1 -d" " -I"HOST" \
    sh -c 'ssh -T HOST uptime'
```

- `-P` Specify the number of processes to start. Usage `nproc` (in coreutil) to get the number of processors.
- `-d` Specify the delimeter to look for in the input.

Processing a large number of files

Sequentially:

```
for file in songs*.wav ; do lame -f $file $file.
```

In parallel:

```
find . -name "songs*.wav" \
    | xargs -I@ -P 5 lame -f @ @.mp3
```

Parallelize grep

If you need to grep though a large directory structure it can be parallelized! Create one process per CPU, looking at 10 files each process:

```
find /some/path -type f \
    | xargs -P$(nproc) -n 10 \
    grep -H 'string-to-search'
```

Convert several files to pdf simultaneously

Sequentially:

```
find ./ -name "*.pdf" \
    | xargs -Istr pdf2ps str
```

In parallel:

```
find ./ -name "*.pdf" \
    | xargs -n 8 -Istr pdf2ps str
```

Output from multiple processes is not buffered

Post-process with `sed`:

```
echo -n 192.168.0.{1..5} \
    | xargs -P5 -n1 -d" " -I"HOST" \
    sh -c 'ssh -T HOST uptime \
        | sed -e "s/^/HOST: /"'
```

Or:

```
echo 1 2 3 \
    | xargs -P3 -n1 sh -c '$PWD/count.sh $* \
    | sed "s/^/$$:/"' - | sort
```

annotate

`annotate` from the `devscripts` package:

```
#!/bin/sh
OUT=`mktemp /tmp/atomic.XXXXXX` || exit 1
ERR=`mktemp /tmp/atomic.XXXXXX` || exit 1
"$@" >> $OUT 2>> $ERR ; EXIT=$?
cat $ERR
cat $OUT
rm -f $OUT $ERR
exit $EXIT
```

xargs is not line-oriented

```
touch important_file
touch 'not important_file'
ls not* | xargs rm
mkdir -p '12" records'
ls | xargs rmdir
```

Run jobs in parallel

```
cat somelist \
    | parallel do_something \
    | process_output
```

- http://www.gnu.org/software/parallel/

Run jobs in parallel

```
cat somelist \
    | parallel do_something \
    | process_output
```

- http://www.gnu.org/software/parallel/
- https://build.opensuse.org/project/repositories?project=home:tange

Run jobs in parallel

```
cat somelist \
    | parallel do_something \
    | process_output
```

- http://www.gnu.org/software/parallel/
- https://build.opensuse.org/project/repositories?project=home:tange
- man parallel

Lightweight job queue

```
echo > jobqueue; tail -f jobqueue | parallel
echo my_command my_args >> jobqueue
```

Run tasks remotely via ssh

```
seq 10 | parallel \
    --sshlogin server1,server2 echo
```

Distribute across many machines

```
parallel --trc {.}.ogg -j+0 \
    -S server1,server2,: \
    'mpg321 -w - {} \
        | oggenc -q0 - -o {.}.ogg' ::: *.mp3
```