Multi-process in shell script

(and when not to do it)

`& $! wait`

Tom Ryder

tom@sanctum.geek.nz

https://sanctum.geek.nz/

Backup script—1/3

- A classic shell scripting task!
- Back up the website files with `rsync`.
- Dump its database with `mariadb-dump`.
- Make sure to catch and report failures, and any error output.

Backup script—2/3

```
#!/usr/bin/sh
dest=/var/lib/backup/$(date -I)
mkdir -p -- "$dest" || exit
cd -- "$dest" || exit
rsync -a /var/www/example.com . || exit
mariadb-dump example >example.sql || exit
```

Backup script—3/3

- Put that in `/usr/local/bin/example-backup`
- Make it executable: `chmod +x`
- Try it out: `sudo example-backup`
- *Success!* Date-stamped backups appear:

```
/var/lib/backup/2026-04-12/example.com
/var/lib/backup/2026-04-12/example.sql
```

Scheduling—1/3

- Right, now let’s set it up to run nightly.
- Schedule it with:
  - your fancy new container manager, or…
  - a systemd timer (recommended), or…
  - plain old cron:

Scheduling—2/3

```
MAILTO=errors@example.com

# Back up example.com every night at 1am
0 1 * * *  root  example-backup
```

Scheduling—3/3

```
/var/lib/backup/2026-04-12/example.com
/var/lib/backup/2026-04-12/example.sql
/var/lib/backup/2026-04-13/example.com
/var/lib/backup/2026-04-13/example.sql
/var/lib/backup/2026-04-14/example.com
/var/lib/backup/2026-04-14/example.sql
…
```

Parallelising—1/3

- The script runs a bit slow, and is mostly CPU-bound. We have multiple CPUs.
- Maybe we could *parallelise* it?
- Could we make the database dump happen *while* the files backup is running?

Parallelising—2/3

- In shell script, you run tasks in the background with an ampersand (&) trailing character:

```
command arguments &
```

- You should then **`wait`** for all such processes to complete before the script itself finishes.

Parallelising—3/3

```
#!/usr/bin/sh
…
rsync -a /var/www/example.com . &
mariadb-dump example >example.sql &
wait
```

Error checking—1/3

- Without any arguments, `wait` will sit there until *all* its background processes have completed.
- Used this way, **it always returns 0, or “success”**.
- If either background process failed, the script doesn’t report failure—not good!
- `wait` on processes *one-by-one* instead, and look at their exit values.

Error checking—2/3

- After backgrounding a process, you can get its **process ID** (PID) from this variable: `$!`
- You can then wait on each process by passing its PID to the `wait` command.
- You can keep the exit value `wait` gives you with this variable: `$?`

Error checking—3/3

```
#!/usr/bin/sh
…
rsync -a /var/www/example.com . & pid1=$!
mariadb-dump example >example.sql & pid2=$!
code=0
wait "$pid1" || code=$?
wait "$pid2" || code=$?
exit "$code"
```

Output—1/2

- If your commands emit standard output or standard error text at the same time, they will be *mixed together* in the same stream, which can be hard to read.
- You may want to write each command to separate log and error files, and then write that back out at the end.

Output—2/2

```
#!/usr/bin/sh
…
rsync -a /var/www/example.com . >log1 2>err1 & pid1=$!
mariadb-dump example >example.sql 2>err2 & pid2=$!
code=0
wait "$pid1" || code=$?
wait "$pid2" || code=$?
cat log?
cat err? >&2
exit "${code:-0}"
```

An easier way—1/2

- That’s how you do it.
  - If you ever really *need* to.
- But in this particular situation? *Don’t bother.*
- Just make *two separate scripts*, and schedule them to run at the same time.
- Easier, safer, clearer.

An easier way—2/2

```
MAILTO=errors@example.com

# Back up example.com every night at 1am
0 1 * * *  root  example-backup-files
0 1 * * *  root  example-backup-mariadb
```

Questions?

- Bash manual—Job control
- Greg's Wiki—ProcessManagement

POSIX (2024) specifications:

- asynchronous commands
- wait command

- **Email:** tom@sanctum.geek.nz
- **Website:** https://sanctum.geek.nz/
- **Fediverse:** @tejr@mastodon.sdf.org