SAS and Makefiles

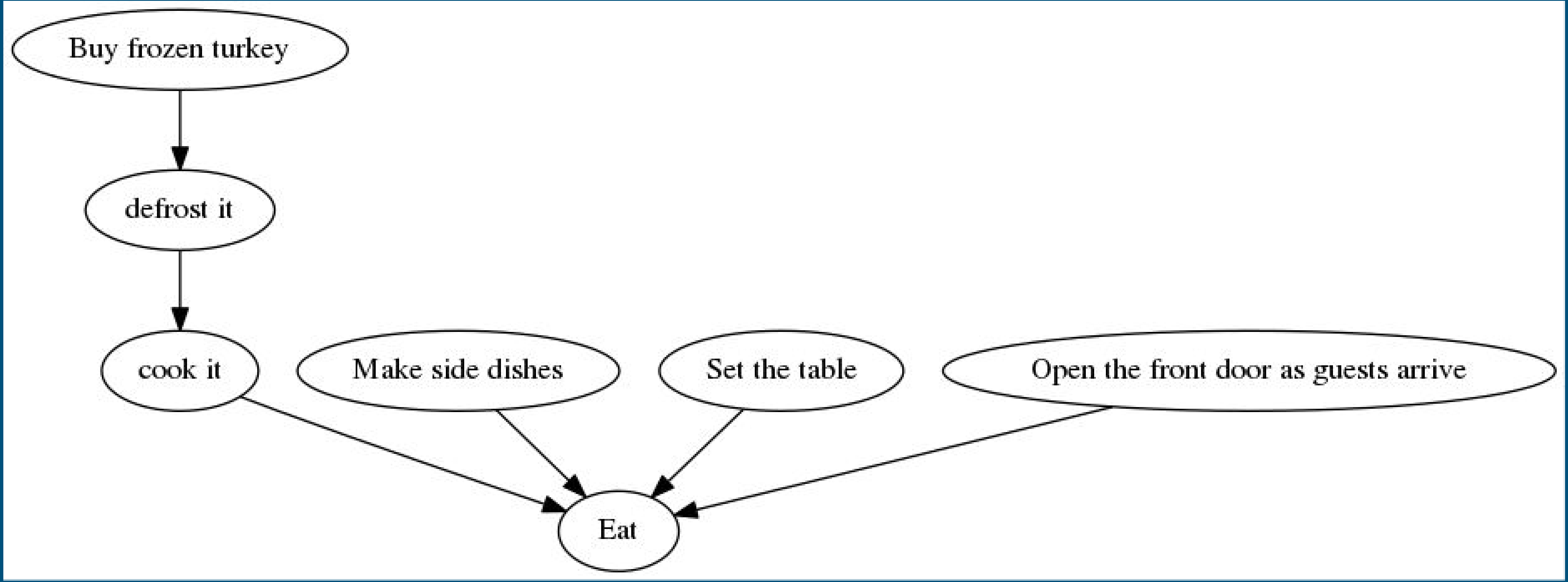
Speed up debugging long-running processes

Analogy: Thanksgiving

1. Buy a frozen turkey, defrost it, cook it
2. Make side dishes
3. Set table
4. Eat!

If something goes wrong at step 3, we don't go out and buy a new turkey!

Thanksgiving dependency tree



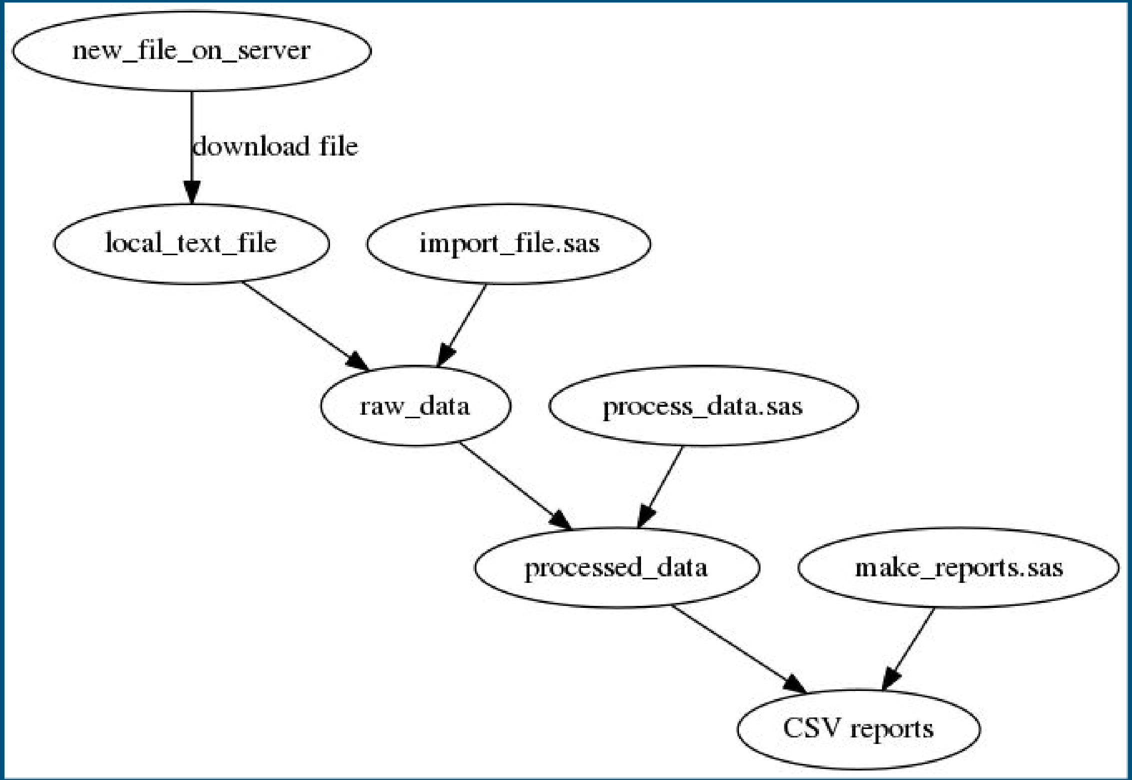
Thanksgiving dependency tree explained

- Dependency trees describe the relationships between:
  - Inputs (e.g., frozen turkey)
  - Outputs (e.g., a thawed turkey)
  - Processes (e.g., defrost)
- If an input is updated, then downstream outputs need to be updated

Contrived SAS workflow

1. Download some gigantic text file.
2. Run **`import_file.sas`** on the downloaded file. That creates a new dataset **`raw_data`**.
3. Run **`process_raw_data.sas`**. It reads **`raw_data`**. And writes out **`processed_data`**.
4. Run **`make_reports.sas`** that reads **`processed_data`** and writes out **`reports.csv`**.

Dependency tree for workflow



Dependency trees and SAS

- Datasets are inputs
- Downloaded text files are inputs
- SAS programs themselves are also inputs
- If I edit a SAS program, I need to remake the outputs of that program!

Makefiles define dependencies

The pattern:

```
Output files: input or input files

    how to transform inputs into outputs
```

Example:

```
raw_data.sas7bdat: /tmp/local_text_file.csv import_file.sas

    sas import_file.sas
```

Makefiles check file timestamps

- Make compares the modified time of the inputs to the outputs.
- Make rebuilds the outputs if:
  - The outputs do not exist
  - The outputs are older than the inputs
- Commands have to return a successful status code!

Very simple Makefile

```
/tmp/local_text_file.csv:
	touch /tmp/local_text_file.csv

clean:
	/bin/rm /tmp/local_text_file.csv
```

Try it out!

```
$ make

touch /tmp/local_text_file.csv

$ make

make: '/tmp/local_text_file.csv' is up to date.
```

- All commands are printed
- Repeat runs don't waste time

Build a particular target

```
$ make clean

/bin/rm /tmp/local_text_file.csv

$ make

touch /tmp/local_text_file.csv

$ make

make: '/tmp/local_text_file.csv' is up to date
```

More neat make stuff

- The --dry-run option shows what make thinks it should do (nice for debugging your Makefile)
- The --jobs option lets you run numerous jobs in parallel (but consider your hardware limits)
- Spend an afternoon reading the make manual page!

```
$ man make
```

216 Software, LLC

We build prototypes of interesting ideas. We're creative, friendly, and reliable.

Matt Wilson, partner

matt@216software.com