MODERN C++ DESIGN PATTERNS

Command-Line Parameters

by Prasanna Ghali

Command-line parameters (1/4)

- When a program is run, it often must be supplied with information
  - May include file name(s) or switches that modify program's behavior
  - This information is called *command-line parameters*
- Example is copy command
  - `cp src-file-name dest-file-name`
- Another example is list directory command
  - `ls` or `ls –a` or `ls -l`

Command-line parameters (2/4)

- To access *command-line parameters*, `main` must have two parameters:

```
int main(int argc, char* argv[]) {
   ...
}
```

- Command-line parameters are called *program parameters* in the C standard.

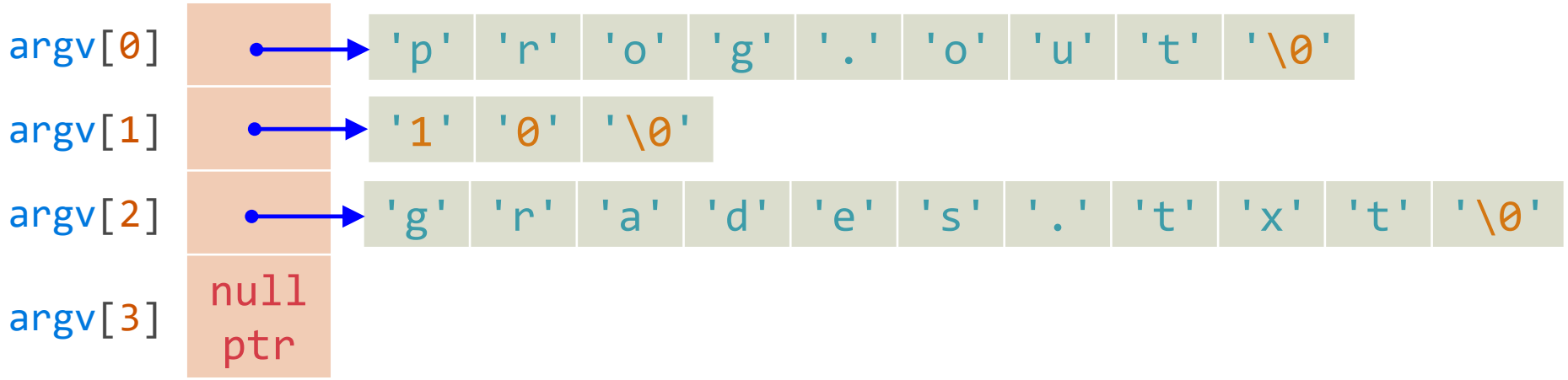
Command-line parameters (3/4)

- `argc` is count of command-line parameters
- `argv` is array of pointers to command-line parameters stored as C-strings
  - `argv[0]` points to program's name
  - `argv[1]` thro' `argv[argc-1]` point to remaining command-line parameters
  - `argv[argc]` always contains *null pointer* that points to nothing

```
int main(int argc, char* argv[]) {
   ...
}
```

Command-line parameters (4/4)

- Assume user executes a program in this manner: *prog.out 10 grades.txt*
  - `argc` equivalent to 3
  - `argv` has type `char* argv[]` with form:



Processing command-line parameters (1/2)

- Iterate over elements in array `argv` using `int` variable as index

```
int main(int argc, char *argv[]) {
  // print command-line parameters
  for (int i{}; i < argc; ++i) {
    std::cout << argv[i] << "\n";
  }
  // other code ...
}
```

Processing command-line parameters (2/2)

- Iterate over elements in `argv` array using variable of type `char**` that initially points to 1st array element

```
int main(int argc, char **argv) {
  // print command-line parameters
  for (char **p = argv; *p; ++p) {
    std::cout << *p << "\n";
  }
}
```