How to List All Commands that a Shell Knows

Background

- A shell knows four kinds of commands.
  1. **Aliases**: these are nicknames for a command with some options. They are defined in the shell's initialization file (`~/.bashrc` for bash, `~/bash_profile` on MacOS).
     - You can list aliases by running the `alias` built-in with no argument.
  2. **Functions**: they are snippets of shell code given a name. Like aliases, they are defined in the shell's initialization file.
     - There is no way to list functions or builtins that works in all shells.
  3. **Builtins**: the shell comes with a small number of built-in commands. Most builtins manipulate the shell state (`cd` changes the current directory, `set` changes options and positional parameters, `export` changes the environment, …). Most shell offer largely the same builtins but each shell has a few extensions to the basic set.
     - You can find a list of builtins in the shell's documentation.
  4. **External commands**: they are independent of the shell. Like other programs, the shell executes external programs by looking them up in the executable search path. The `PATH` environment variable contains a colon-separated list of directories to search for programs.
- **In case there are commands of several types by the same name, the first match in the order above is executed.**

Methods

1. You can use the `compgen` command in bash:

```
compgen -c # will list all the commands you could run.
compgen -a # will list all the aliases you could run.
compgen -b # will list all the built-ins you could run.
compgen -k # will list all the keywords you could run.
compgen -A function # will list all the functions you could run.
compgen -A function -abck # will list all the above in one go.
```

2. Custom Script - The following shell-agnostic snippet lists all available external programs:

```
case "$PATH" in
  (*[!:]:) PATH="$PATH:" ;;
esac
```

```
set -f; IFS=:
for dir in $PATH; do
  set +f
  [ -z "$dir" ] && dir="."
  for file in "$dir"/*; do
    if [ -x "$file" ] && ! [ -d "$file" ]; then
      printf '%s = %s\n' "${file##*/}" "$file"
    fi
  done
done
```

- Note there is an edge case in Bash: Hashed Commands...
- Bash Reference Manual says: "A full search of the directories in $PATH is performed only if the command is not found in the hash table" [2]

References

1. https://unix.stackexchange.com/questions/94775/list-all-commands-that-a-shell-knows
2. http://www.gnu.org/software/bash/manual/bash.html#Command-Search-and-Execution