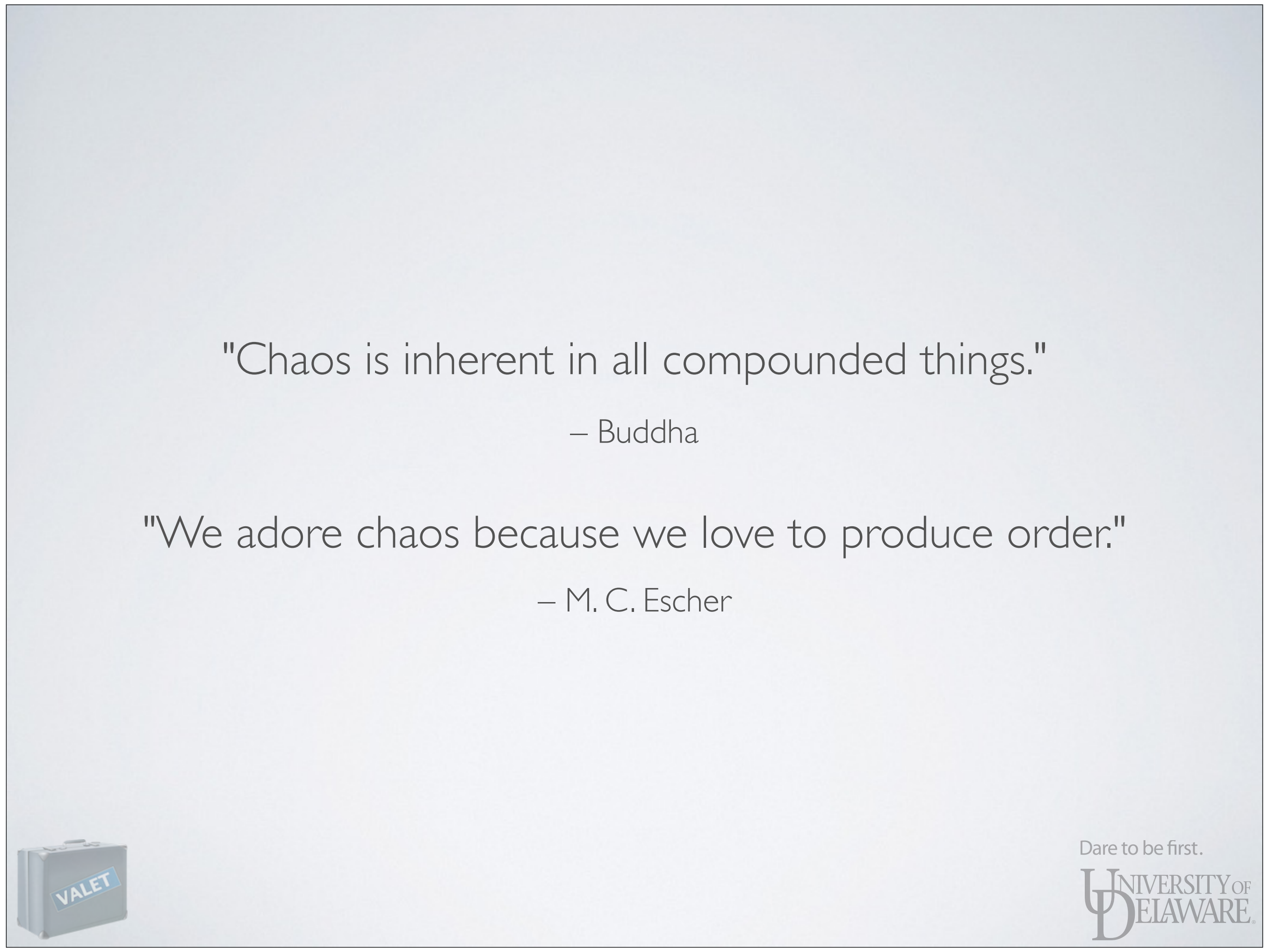
MODULAR ENVIRONMENT MANAGEMENT WITH VALET

Dr. Jeffrey Frey
University of Delaware, IT

GOALS AS QUESTIONS

- Why is environment management necessary?
- How does VALET help manage the environment?
- Can I use VALET to manage my own software installs?



"Chaos is inherent in all compounded things."

– Buddha

"We adore chaos because we love to produce order."

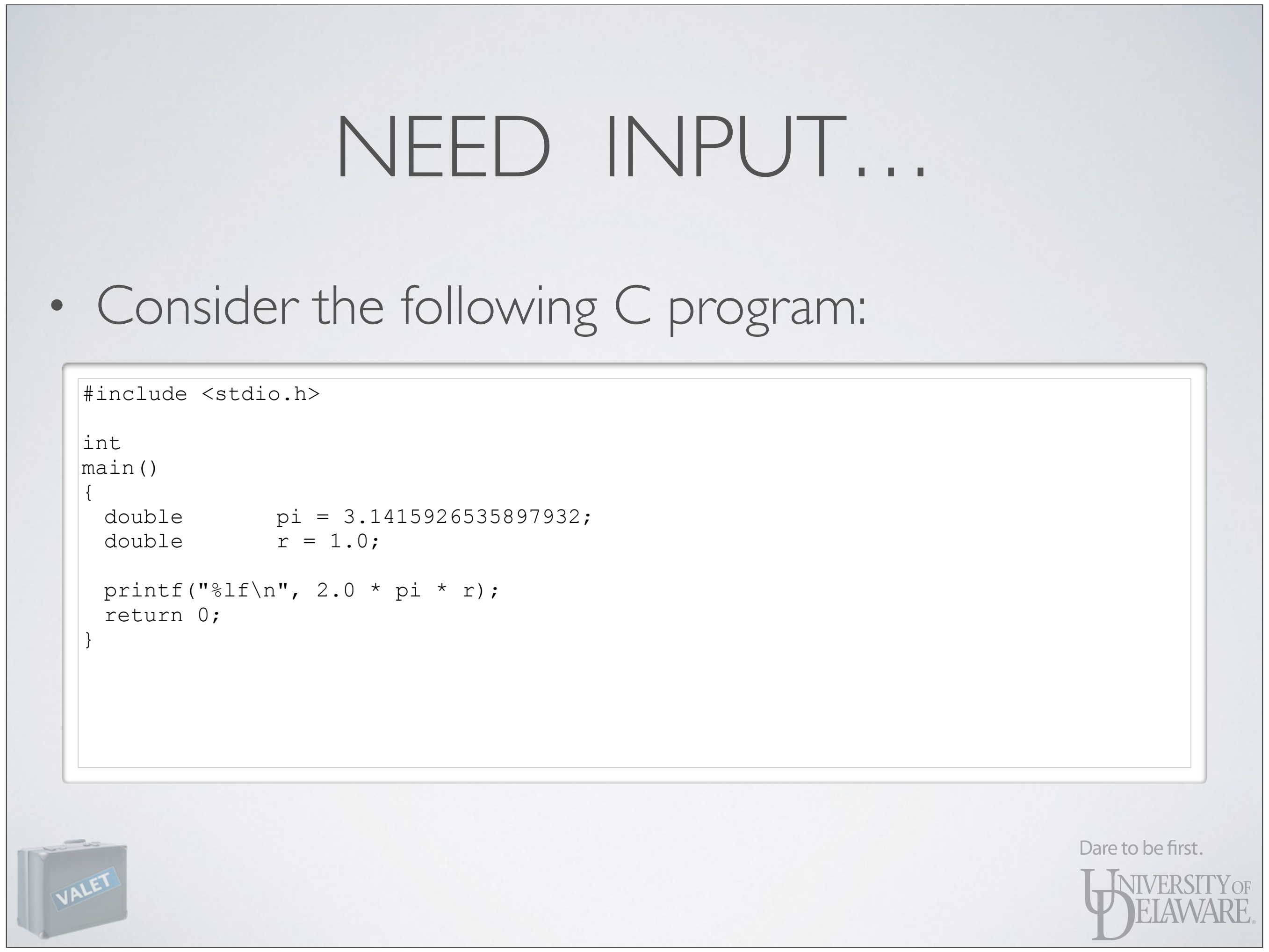
– M. C. Escher

VALET

Dare to be first.

UNIVERSITY OF DELAWARE

- Very true in the realm of computers: a system comprised of many different components designed by different individuals with different goals and different ideas of how best to solve a problem.
- A sysadmin spends a lot of time determining how to make those different parts work together in harmony: organize the chaos.



NEED INPUT…

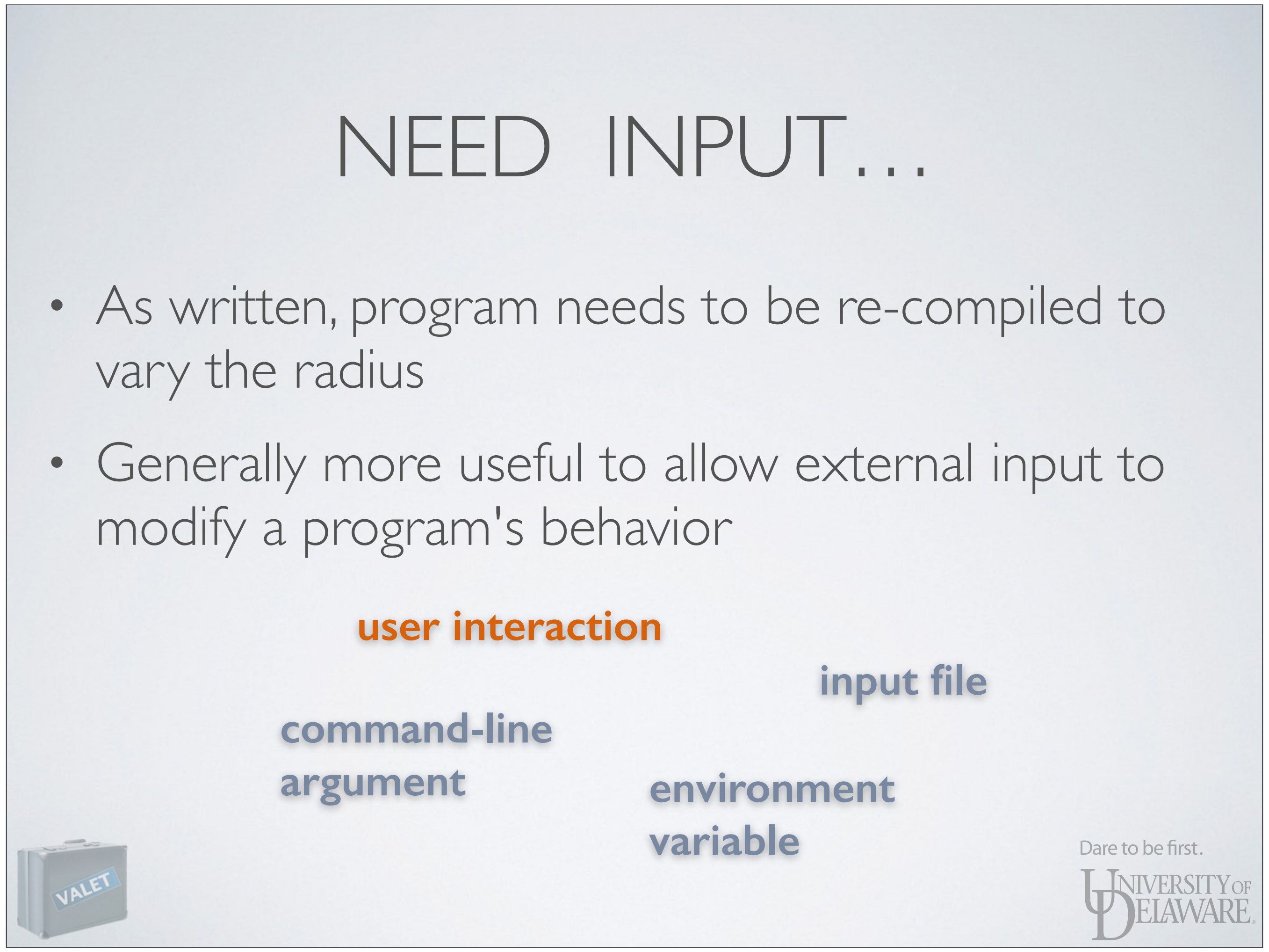
- Consider the following C program:

```
#include <stdio.h>

int
main()
{
  double       pi = 3.1415926535897932;
  double       r = 1.0;

  printf("%lf\n", 2.0 * pi * r);
  return 0;
}
```

Dare to be first.

UNIVERSITY OF DELAWARE

- What does this program do?
- Can the behavior of this program be influenced externally? In other words, how do I go about altering the calculation?



NEED INPUT…

- As written, program needs to be re-compiled to vary the radius
- Generally more useful to allow external input to modify a program's behavior

user interaction

input file

command-line argument

environment variable

Dare to be first.

- What kinds of external input might be used?
- User interaction used to be more prevalent but tends to be avoided today — why?

NEED INPUT…

- Command-line arguments

```
#include <stdio.h>
#include <stdlib.h>

int
main(
  int           argc,
  const char*  argv[]
)
{
  double       pi = 3.1415926535897932;
  double       r = 1.0;

  if ( argc > 1 ) r = strtod(argv[1], NULL);
  printf("%lf\n", 2.0 * pi * r);
  return 0;
}
```

NEED INPUT…

- Command-line arguments

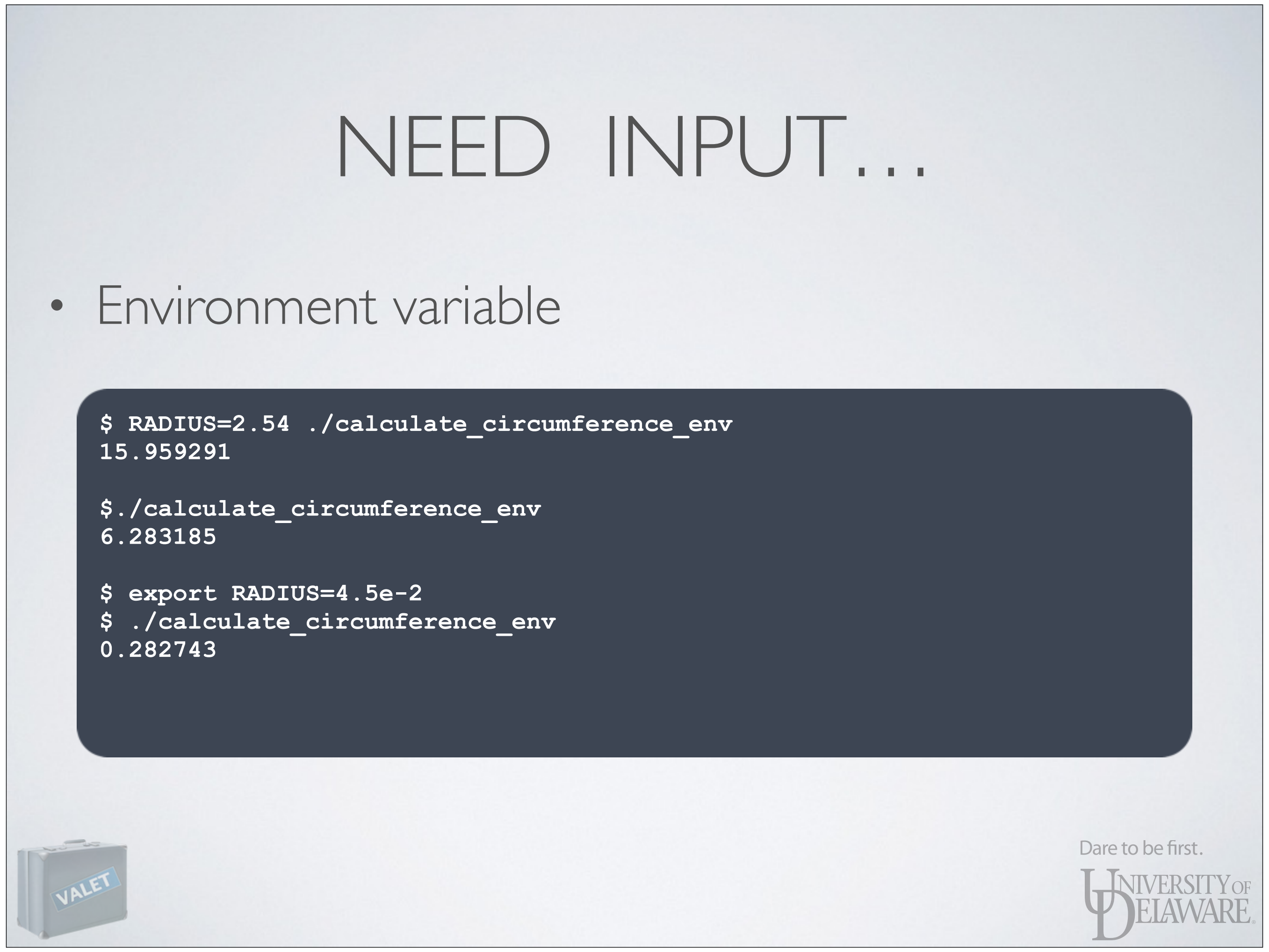
```
$ ./calculate_circumference 2.54
15.959291

$ ./calculate_circumference 4.5e-2
0.282743
```

Dare to be first.

NEED INPUT…

- Environment variable

```
#include <stdio.h>
#include <stdlib.h>

int
main()
{
  double       pi = 3.1415926535897932;
  double       r = 1.0;
  char         *r_env = getenv("RADIUS");

  if ( r_env ) r = strtod(r_env, NULL);
  printf("%lf\n", 2.0 * pi * r);
  return 0;
}
```



NEED INPUT…

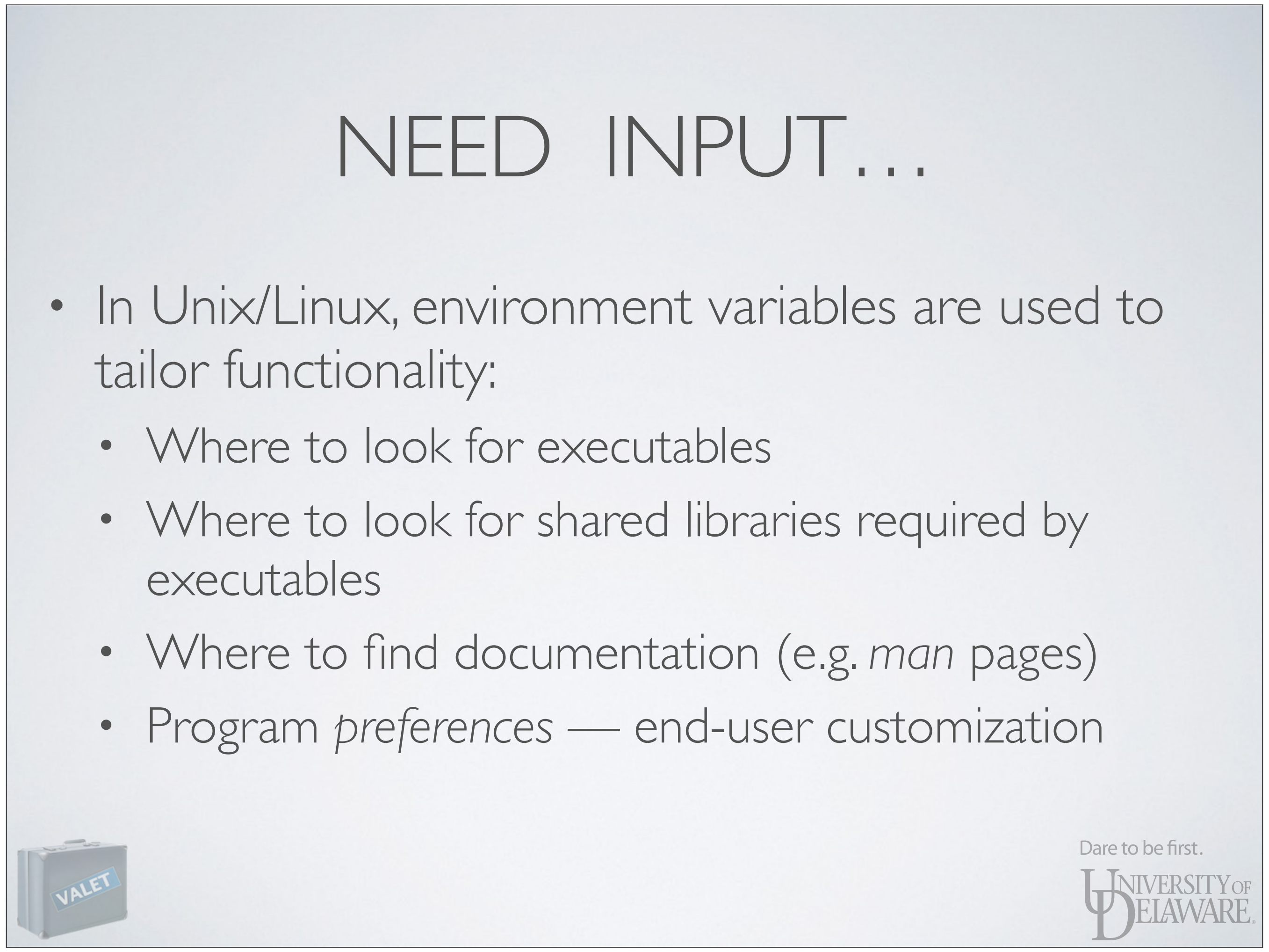
- Environment variable

```
$ RADIUS=2.54 ./calculate_circumference_env
15.959291

$./calculate_circumference_env
6.283185

$ export RADIUS=4.5e-2
$ ./calculate_circumference_env
0.282743
```

Dare to be first.

UNIVERSITY OF DELAWARE

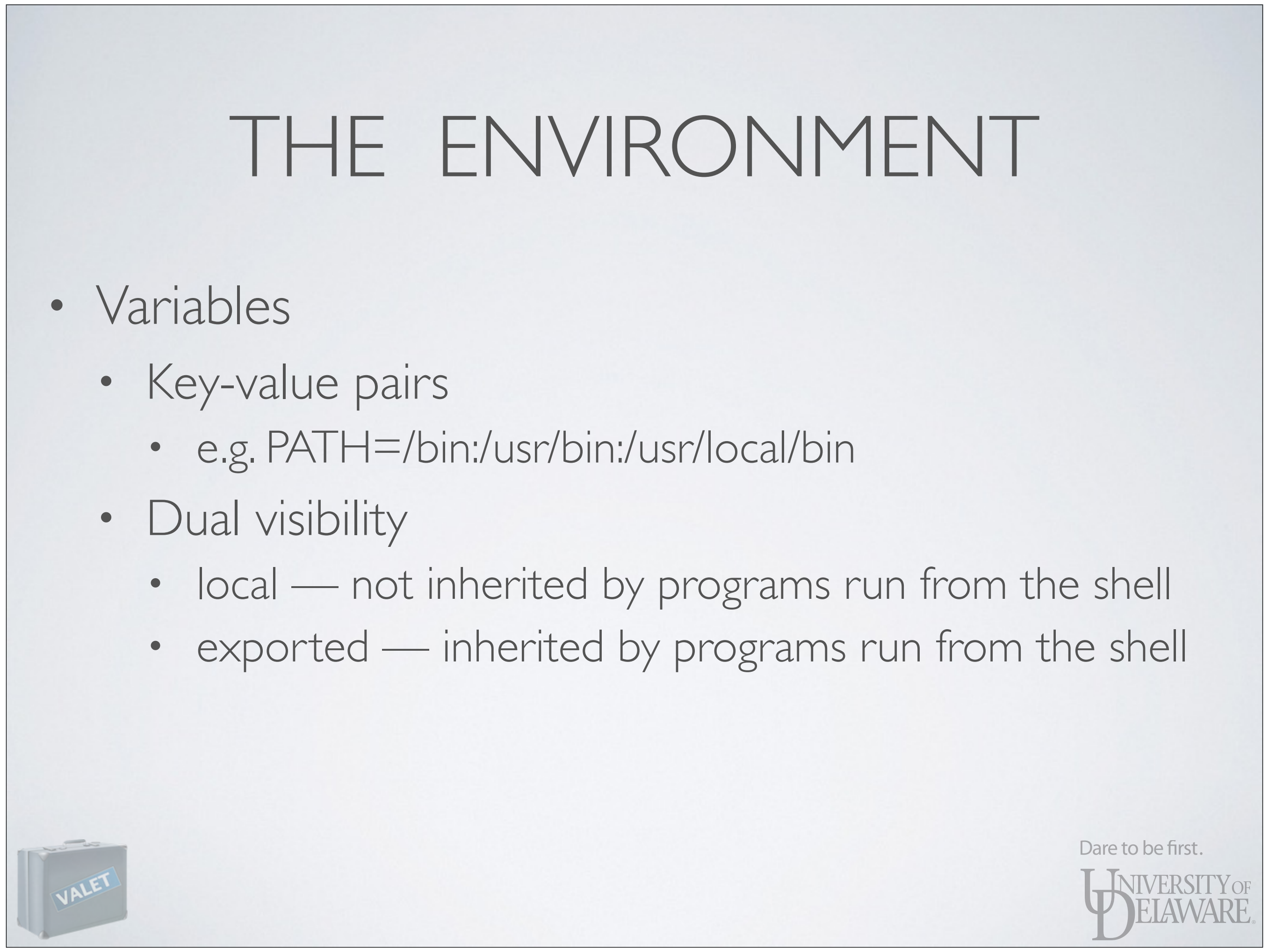
- Note the syntax used in first line: set environment variable RADIUS in the context of the program being executed, NOT in the shell itself.



NEED INPUT…

- In Unix/Linux, environment variables are used to tailor functionality:
  - Where to look for executables
  - Where to look for shared libraries required by executables
  - Where to find documentation (e.g. *man* pages)
  - Program *preferences* — end-user customization

Dare to be first.



THE ENVIRONMENT

- Variables
  - Key-value pairs
    - e.g. PATH=/bin:/usr/bin:/usr/local/bin
  - Dual visibility
    - local — not inherited by programs run from the shell
    - exported — inherited by programs run from the shell

Dare to be first.

UNIVERSITY OF DELAWARE

- The environment is more than just variables, though

THE ENVIRONMENT

- Variables
- Aliases
  - Shortcut for a longer command
    - e.g. "l." ➔ "ls -d .*"

THE ENVIRONMENT

- Variables
- Aliases
- Functions
  - A sequence of shell commands identified by a name
  - May accept a list of arguments, just like a program

THE ENVIRONMENT

- Variables
- Aliases
- Functions

Exported variables span all shells and programs.

Local variables, aliases, and functions are features of the shell itself.

SOFTWARE AND ENVIRONMENT

- Have you ever seen something like this in software documentation?

To begin using myProgram, edit your .bashrc file and add these lines at the end:

```
export PATH=~/myProgram/bin:$PATH
export LD_LIBRARY_PATH=~/myProgram/lib:$LD_LIBRARY_PATH
```

SOFTWARE AND ENVIRONMENT

- Making such changes to your shell login files may have unintended side effects.
  - Each time you login with ssh, those changes are applied to the shell.
  - Each job you submit, when run, has those changes applied to its shell.
  - In other words, such changes are **global** in scope

Dare to be first.

SOFTWARE AND ENVIRONMENT

```
[user@farber ~]$ myProgram
This is version 1 of myProgram.

[user@farber ~]$ qsub
date
myProgram

^D
Your job 5393[illegible] ("STDIN") has been submitted
[user@farber ~
```

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

PATH=$HOME/bin:/opt/sbin:$PATH
export PATH

# User specific aliases and functions
PATH=$HOME/version2:$PATH
```

Dare to be first.

UNIVERSITY OF DELAWARE

- It's possible you could unknowingly sabotage your own running jobs
    - E.g. you submit a job to use version 1 of myProgram
    - Before that job executes, you install version 2 and change .bashrc to point to it
    - When your job executes, it uses version 2 when you wanted it to use version 1

SOFTWARE AND ENVIRONMENT

```
[user@farber ~]$ cat STDIN.o53935
Mon Apr 13 13:37:11 EDT 2015
This is version 2 of myProgram.

[user@farber ~]$
```

Dare to be first.

SOFTWARE AND ENVIRONMENT

- Making such changes to your shell login files may have unintended side effects.
- Places the burden squarely on you
  - YOU must know **how** to make changes
  - YOU must know **what** to add to $PATH, etc.
  - YOU must keep track of dependencies
  - YOU must debug any problems that arise due to interplay between packages

Dare to be first.
University of Delaware

SOFTWARE AND ENVIRONMENT

- In short, global changes added to shell login files are only appropriate for modifying how the shell itself behaves
  - Aliases for often-used commands
  - Functions in lieu of scripts for some tasks
  - Standard variables (e.g. EDITOR)
    - Even okay to alter PATH, e.g. add "$HOME/bin"

Dare to be first.

UNIVERSITY OF DELAWARE.

SOFTWARE AND ENVIRONMENT

- Global changes made to login files are appropriate for modifying how the shell itself behaves
- So…what to do about software packages?

Dare to be first.

- What's the opposite of GLOBAL?
- You'd like to make changes LOCAL to the individual shell.

SOFTWARE AND ENVIRONMENT

- Need a program that can:
  - ✓ Model a *package* and one or more *versions* of it
    - Paths to executables, libraries, documentation
    - Dependencies on other packages
    - Incompatibilities with other packages
    - Changes to environment variables

SOFTWARE AND ENVIRONMENT

- Need a program that can:
  - ✓ Model a *package* and one or more *versions* of it
  - ✓ Make changes to the environment
    - Check for incompatibilities
    - Recursively add any dependencies
    - Perform actions:
      - add executable paths to $PATH, library paths to $LD_LIBRARY_PATH, etc.
      - alter other environment variables, aliases, functions

Dare to be first.

UNIVERSITY OF DELAWARE.

SOFTWARE AND ENVIRONMENT

- Need a program that can:
  - ✓ Model a *package* and one or more *versions* of it
  - ✓ Make changes to the environment
  - ✓ Revert changes
    - Create a "snapshot" of environment prior to changes
    - Restore a "snapshot"

Dare to be first.

ENVIRONMENT MODULES

- One solution is the *environment modules* program
  - Implemented as TCL scripting language commands
  - Version of a package = a TCL script

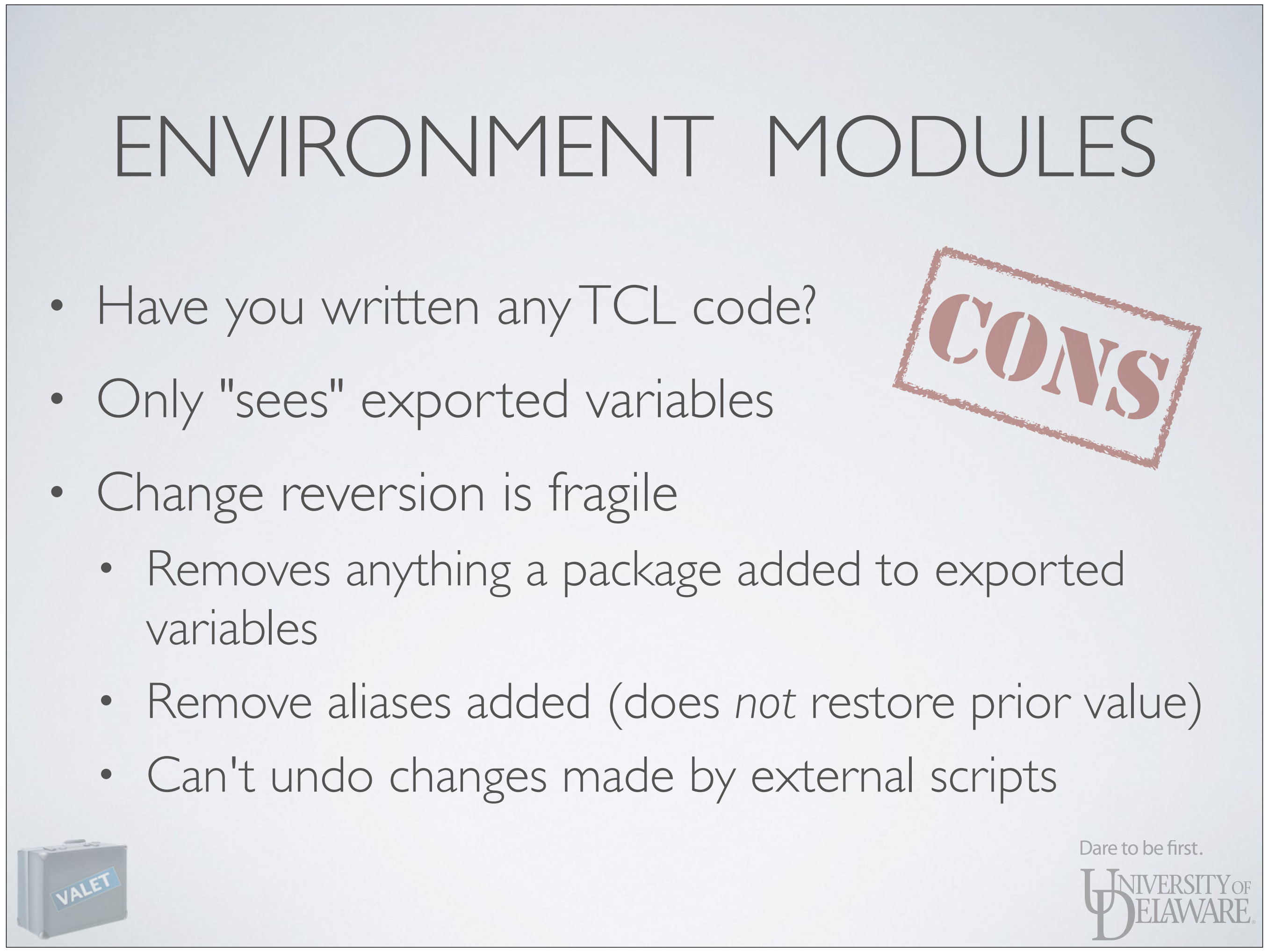
ENVIRONMENT MODULES

- Present on many HPC systems — might call it the *de facto* standard
- Some software vendors provide module files for their software
- Relatively straightforward

PROS

Dare to be first.

UNIVERSITY OF DELAWARE



ENVIRONMENT MODULES

CONS

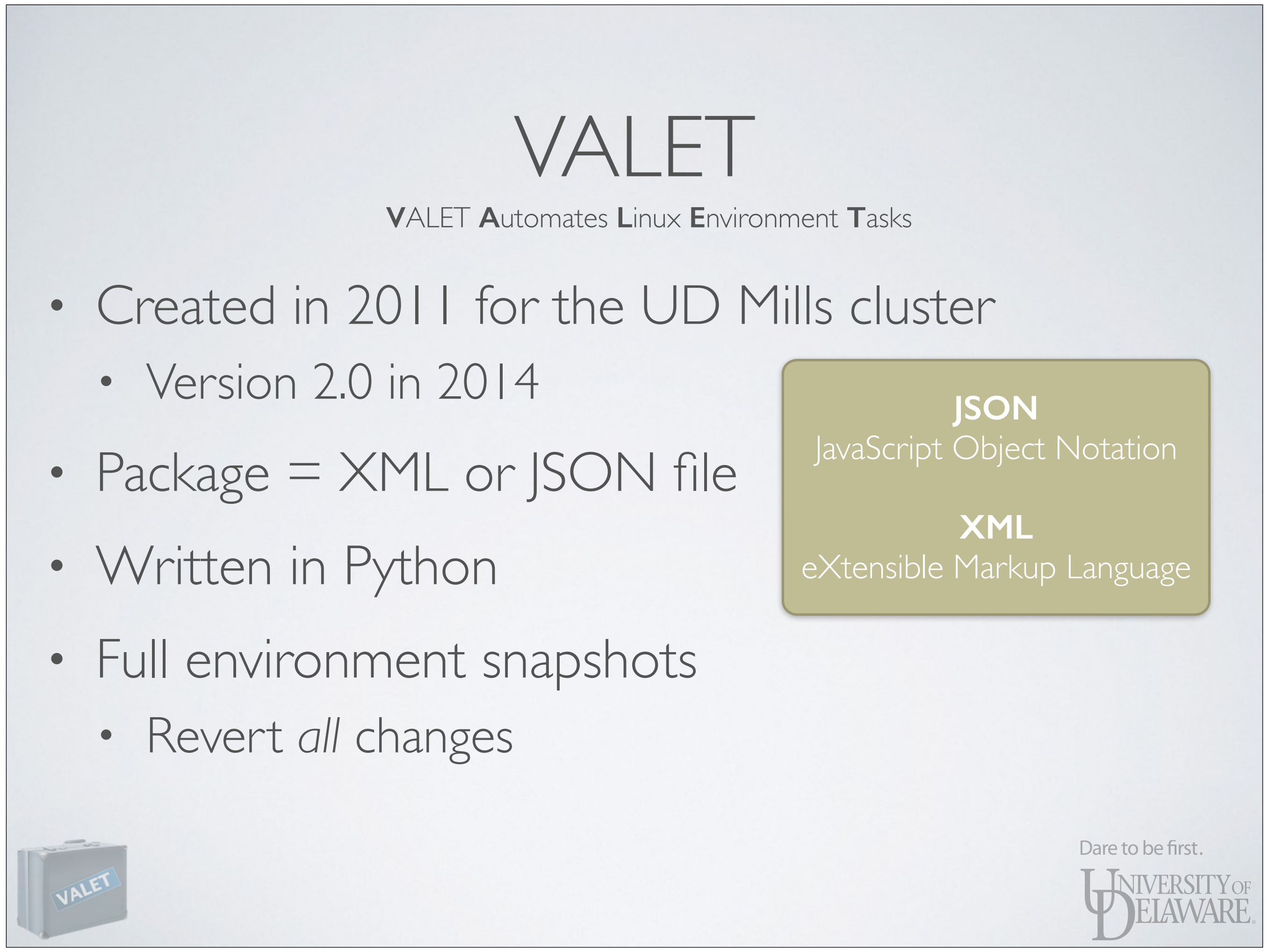
- Have you written any TCL code?
- Only "sees" exported variables
- Change reversion is fragile
  - Removes anything a package added to exported variables
  - Remove aliases added (does *not* restore prior value)
  - Can't undo changes made by external scripts

VALET

Dare to be first.

University of Delaware

-



VALET

VALET **A**utomates **L**inux **E**nvironment **T**asks

- Created in 2011 for the UD Mills cluster
  - Version 2.0 in 2014
- Package = XML or JSON file
- Written in Python
- Full environment snapshots
  - Revert *all* changes

JSON
JavaScript Object Notation

XML
eXtensible Markup Language

Dare to be first.
UNIVERSITY OF DELAWARE

-

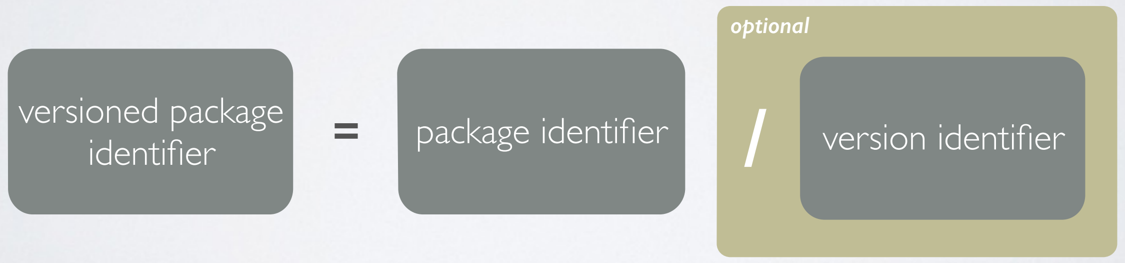
PACKAGE IDENTIFIERS

- A *package* is identified by a string with the following conditions:
  - It must start with a letter or number
  - If can contain zero or more additional letters, numbers, underscore, dot, plus, or hyphen
  - As a regular expression:

```
[a-z0-9][a-z0-9_.+-]*
```

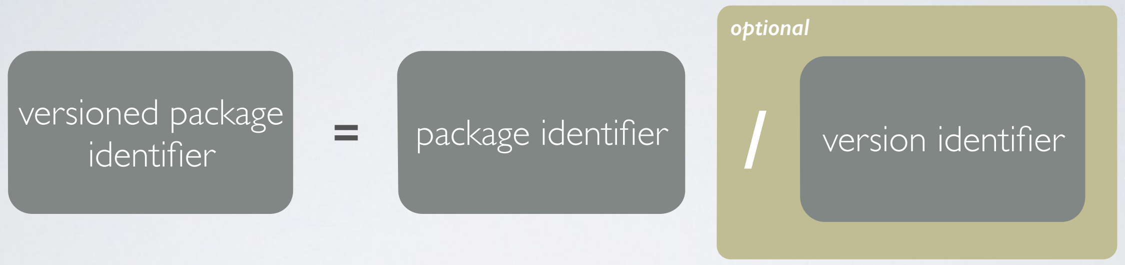
VERSIONED PACKAGE IDENTIFIERS

- A *version* is identified by a string with the same conditions as a *package*
- A *versioned package* identifier is the combination of the two:



Dare to be first.

VERSIONED PACKAGE IDENTIFIERS

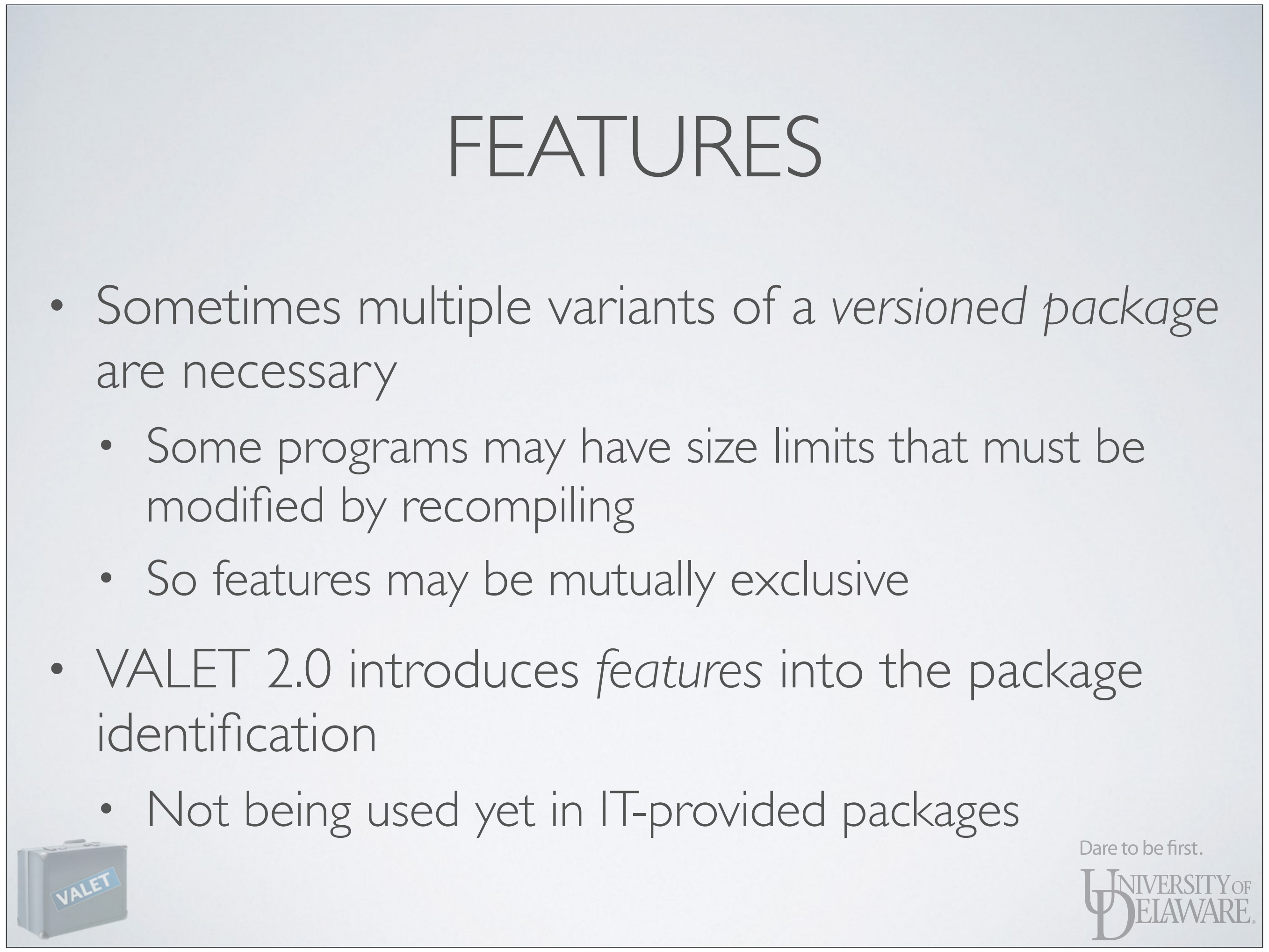


- The version id *default* is reserved
  - Corresponds to whatever version of a package is marked as the default version
- Omitting the version identifier = implied *default*

VERSIONED PACKAGE IDENTIFIERS

identifier	meaning
`gaussian`	Gaussian quantum chemistry software, default version
`gaussian/default`	Gaussian quantum chemistry software, default version
`gaussian/g09a02`	Gaussian '09, revision A02
`gaussian/g03e01`	Gaussian '03, revision E01

Dare to be first.
UNIVERSITY OF DELAWARE



FEATURES

- Sometimes multiple variants of a *versioned package* are necessary
  - Some programs may have size limits that must be modified by recompiling
  - So features may be mutually exclusive
- VALET 2.0 introduces *features* into the package identification
  - Not being used yet in IT-provided packages

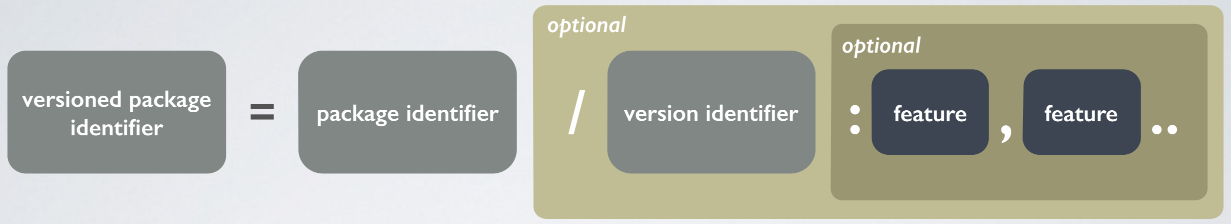
VALET

Dare to be first.

UNIVERSITY OF DELAWARE

- Stronger documentation and testing is required before feature support is considered "fully baked"
- Mills still using VALET 1.0 so this is not available there
- Feel free to use it in package definitions that you create!

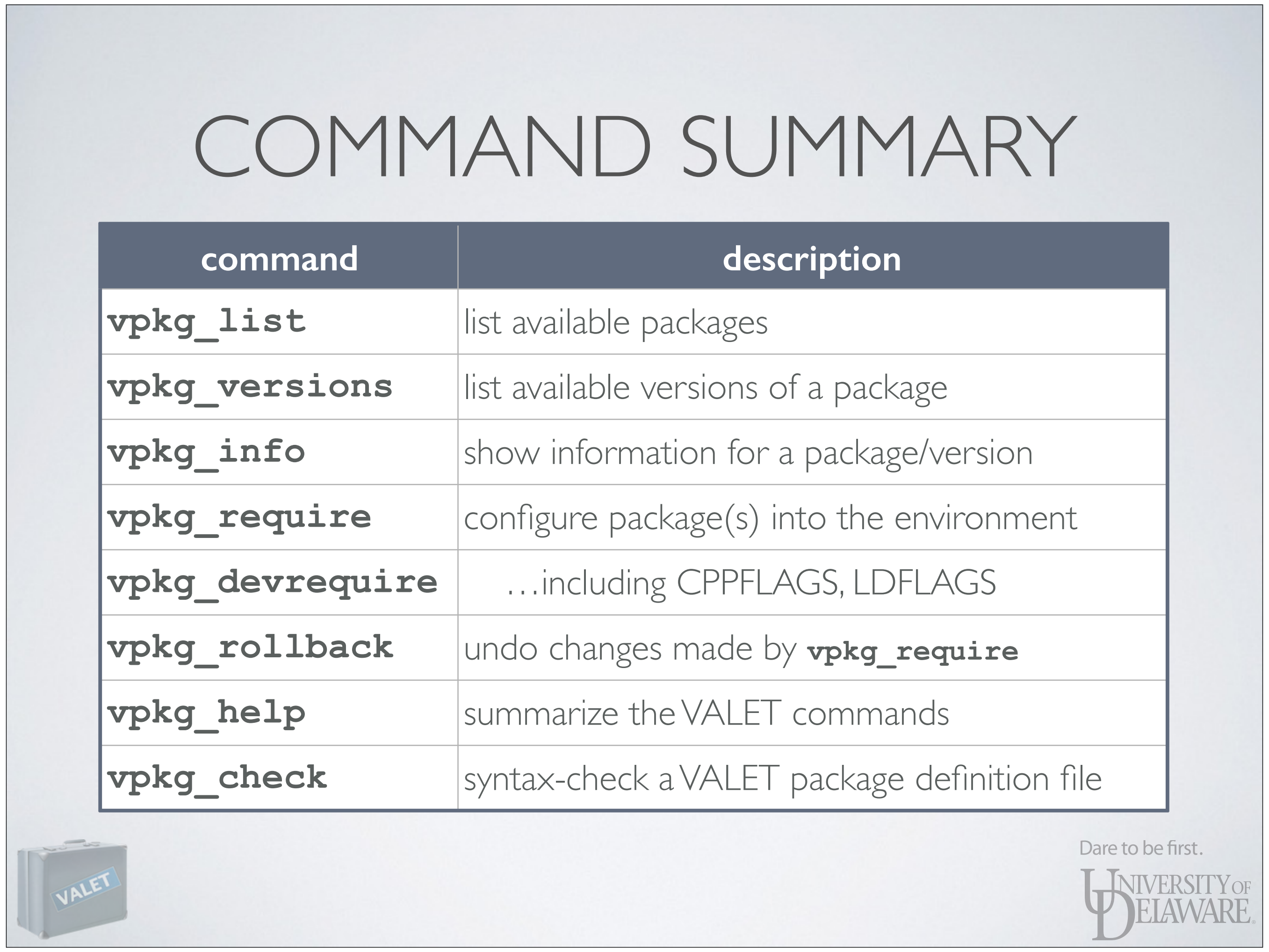
FEATURES



- A list of one or more *features* may be appended to the versioned package identifier using a colon
  - Comma-delimited, no significance to order
  - Same format as package, version

FEATURES

identifier	meaning
`acml/6.0.5.7:gcc`	ACML 6.0 for GCC compilers
`acml/6.0.5.7:intel,openmp`	ACML 6.0 for Intel compilers; OpenMP parallelism
`acml/6.0.5.7:openmp,intel`	Same as previous — feature order does not matter
`openfoam/2.3.0:gcc,dp,opt`	OpenFOAM 2.3.0 compiled with GCC; double-precision; w/ compiler optimizations
`openfoam/2.3.0:gcc,sp,debug`	OpenFOAM 2.3.0 compiled with GCC; single-precision; w/o compiler optimizations



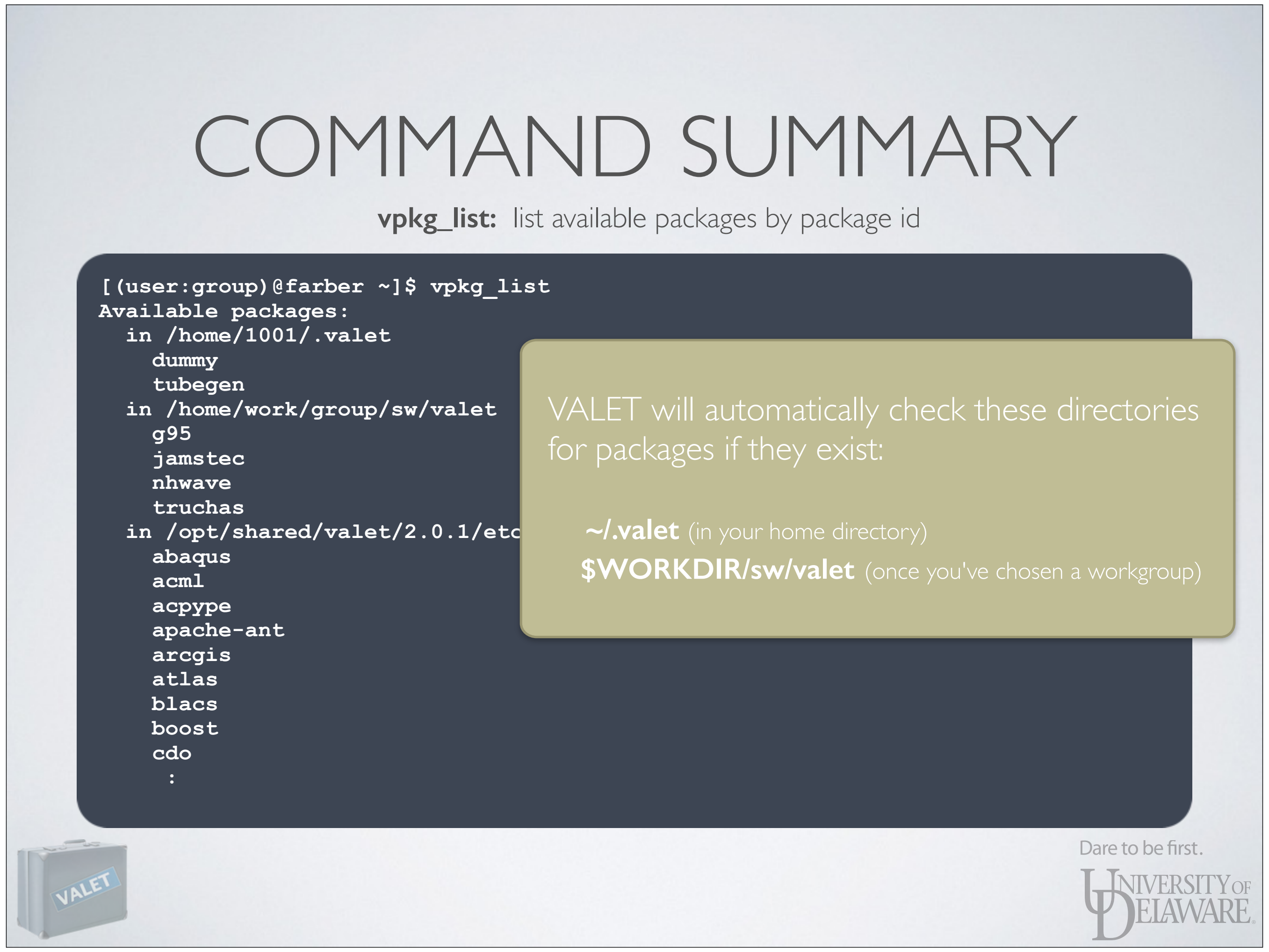
COMMAND SUMMARY

command	description
`vpkg_list`	list available packages
`vpkg_versions`	list available versions of a package
`vpkg_info`	show information for a package/version
`vpkg_require`	configure package(s) into the environment
`vpkg_devrequire`	…including CPPFLAGS, LDFLAGS
`vpkg_rollback`	undo changes made by **`vpkg_require`**
`vpkg_help`	summarize the VALET commands
`vpkg_check`	syntax-check a VALET package definition file

Dare to be first.

UNIVERSITY OF DELAWARE.

-



COMMAND SUMMARY

vpkg_list: list available packages by package id

```
[(user:group)@farber ~]$ vpkg_list
Available packages:
  in /home/1001/.valet
    dummy
    tubegen
  in /home/work/group/sw/valet
    g95
    jamstec
    nhwave
    truchas
  in /opt/shared/valet/2.0.1/etc
    abaqus
    acml
    acpype
    apache-ant
    arcgis
    atlas
    blacs
    boost
    cdo
     :
```

VALET will automatically check these directories for packages if they exist:

~/.valet (in your home directory)

$WORKDIR/sw/valet (once you've chosen a workgroup)

Dare to be first.

UNIVERSITY OF DELAWARE

- If you want to use your own VALET package files in addition to what IT provides, create a directory named "dot valet" in your home directory and put them in there.
- Or, if you're building software for your workgroup, use the "sw/valet" directory.

COMMAND SUMMARY

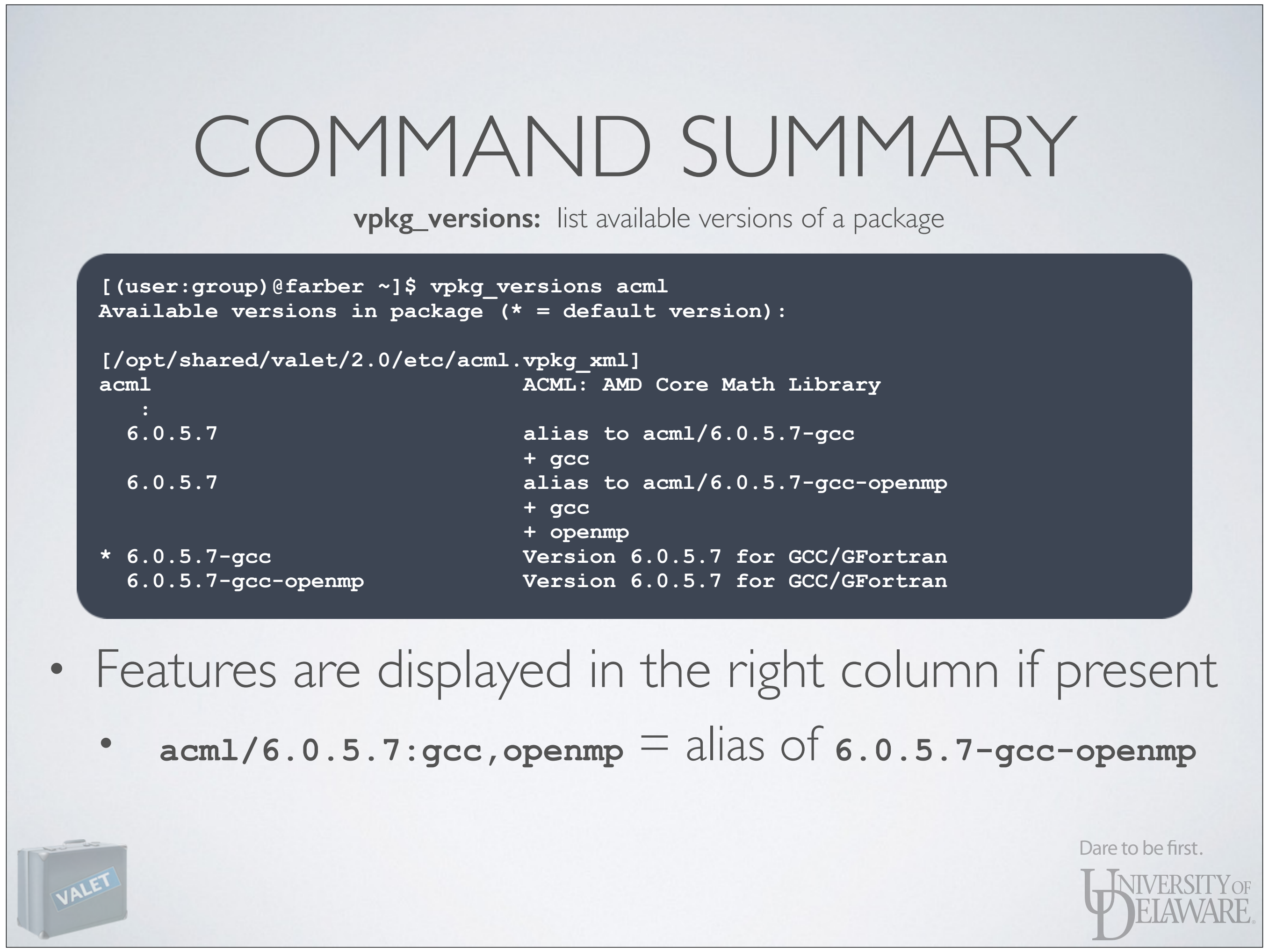
vpkg_versions: list available versions of a package

```
[(user:group)@farber ~]$ vpkg_versions openmpi
Available versions in package (* = default version):

[/opt/shared/valet/2.0/etc/openmpi.vpkg_json]
openmpi            Open MPI:  Message-Passing Interface
  1.8              alias to openmpi/1.8.2
* 1.8.2            Version 1.8.2, with GCC(system) compilers
  1.8.2-gcc-4.8.3  Version 1.8.2, with GCC(4.8.3) compilers
  1.8.2-intel64    Version 1.8.2, with Intel64(2015) compilers
  gcc              alias to openmpi/1.8.2
  intel64          alias to openmpi/1.8.2-intel64
```

- Shows the path to the *package definition file*
- Shows the default version
- Shows all version ids as well as *version aliases*

Dare to be first.
UNIVERSITY OF DELAWARE

-



COMMAND SUMMARY

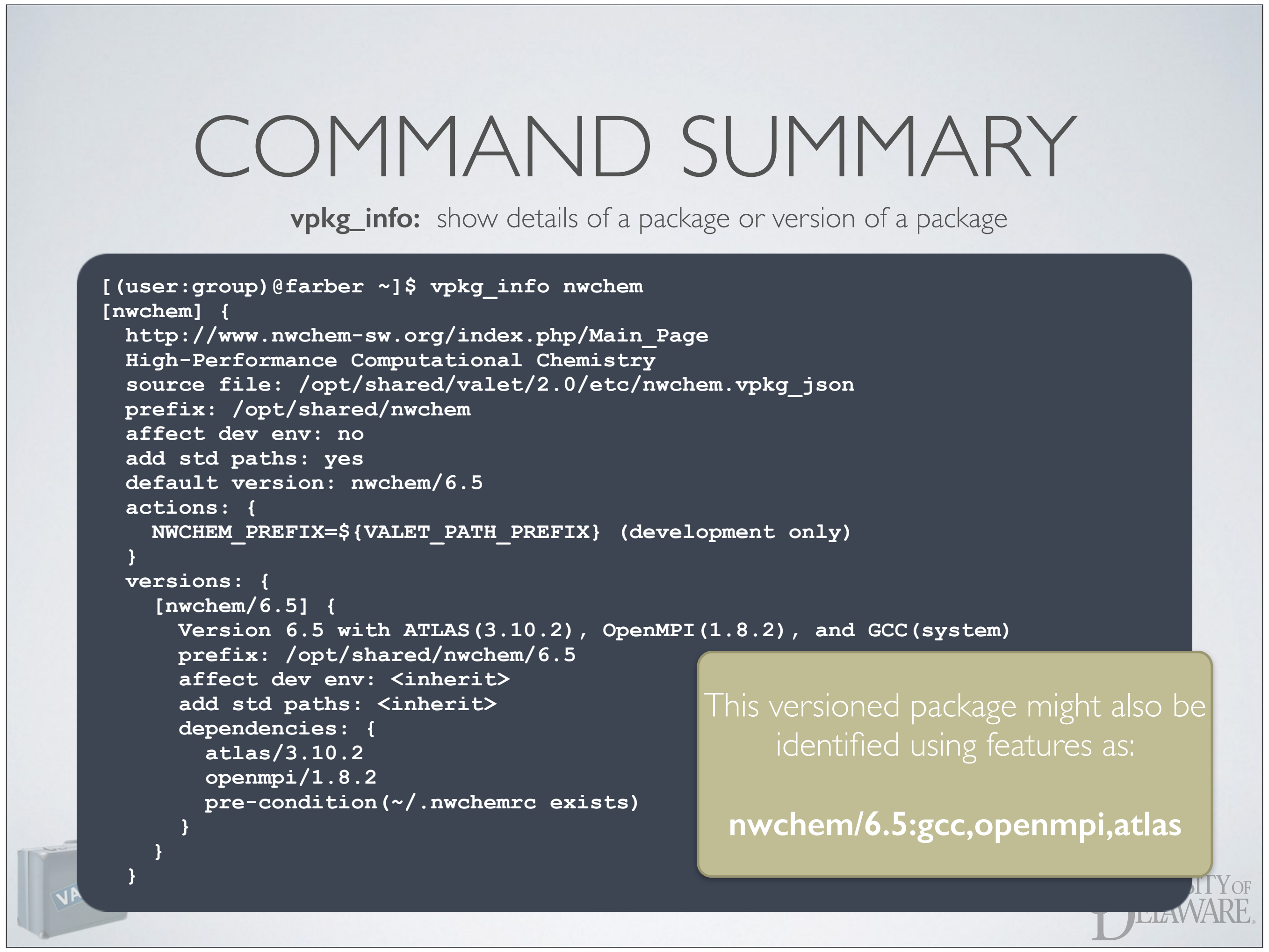
vpkg_versions: list available versions of a package

```
[(user:group)@farber ~]$ vpkg_versions acml
Available versions in package (* = default version):

[/opt/shared/valet/2.0/etc/acml.vpkg_xml]
acml                                ACML: AMD Core Math Library
   :
  6.0.5.7                           alias to acml/6.0.5.7-gcc
                                    + gcc
  6.0.5.7                           alias to acml/6.0.5.7-gcc-openmp
                                    + gcc
                                    + openmp
* 6.0.5.7-gcc                       Version 6.0.5.7 for GCC/GFortran
  6.0.5.7-gcc-openmp                Version 6.0.5.7 for GCC/GFortran
```

- Features are displayed in the right column if present
  - **`acml/6.0.5.7:gcc,openmp`** = alias of **`6.0.5.7-gcc-openmp`**

Dare to be first.

-



COMMAND SUMMARY

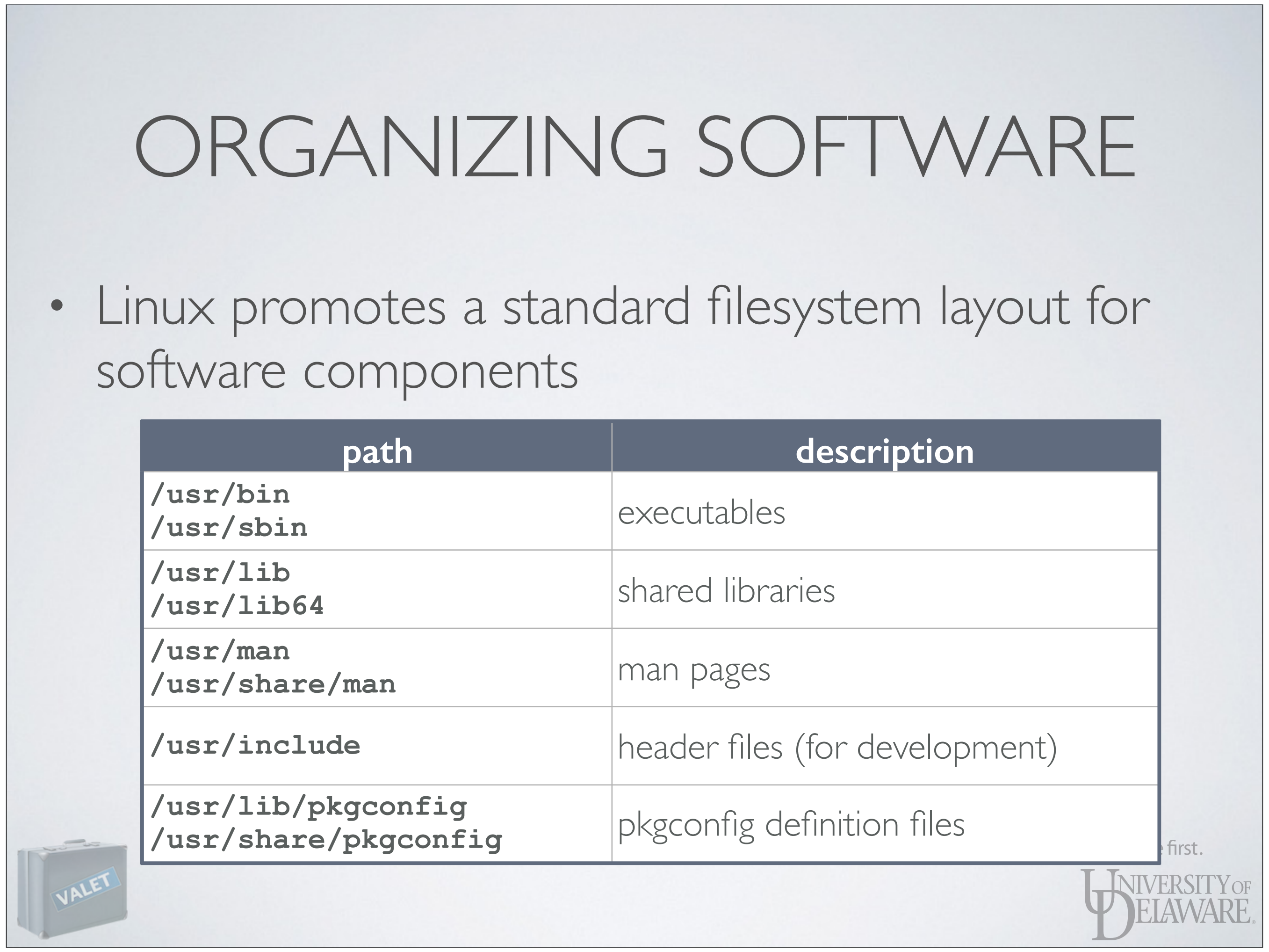
vpkg_info: show details of a package or version of a package

```
[(user:group)@farber ~]$ vpkg_info nwchem
[nwchem] {
  http://www.nwchem-sw.org/index.php/Main_Page
  High-Performance Computational Chemistry
  source file: /opt/shared/valet/2.0/etc/nwchem.vpkg_json
  prefix: /opt/shared/nwchem
  affect dev env: no
  add std paths: yes
  default version: nwchem/6.5
  actions: {
    NWCHEM_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [nwchem/6.5] {
      Version 6.5 with ATLAS(3.10.2), OpenMPI(1.8.2), and GCC(system)
      prefix: /opt/shared/nwchem/6.5
      affect dev env: <inherit>
      add std paths: <inherit>
      dependencies: {
        atlas/3.10.2
        openmpi/1.8.2
        pre-condition(~/.nwchemrc exists)
      }
    }
  }
```

This versioned package might also be identified using features as:

nwchem/6.5:gcc,openmpi,atlas

- Prefix: directory wherein the one-or-more versions of nwchem are installed
- Actions: the modifications to be made to the environment
- Dependencies: other packages which must be present for this one to work; tests which must be satisfied
- Standard paths: look for directories like "bin" and "lib" and automatically add them to the appropriate environment variables (PATH and LD_LIBRARY_PATH). But what are "standard paths?"



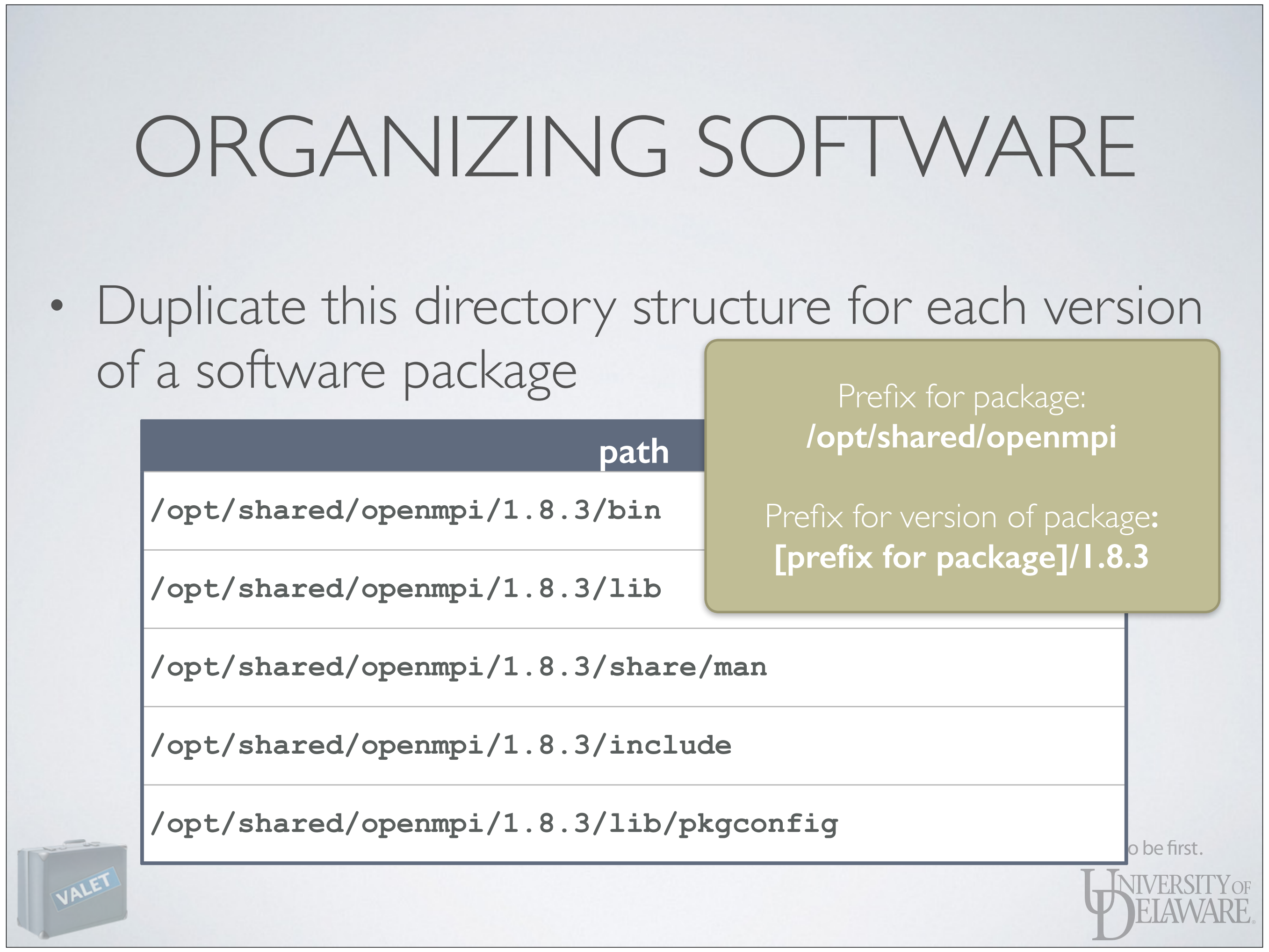
ORGANIZING SOFTWARE

- Linux promotes a standard filesystem layout for software components

path	description
`/usr/bin` `/usr/sbin`	executables
`/usr/lib` `/usr/lib64`	shared libraries
`/usr/man` `/usr/share/man`	man pages
`/usr/include`	header files (for development)
`/usr/lib/pkgconfig` `/usr/share/pkgconfig`	pkgconfig definition files

first.

- Some of these directories are also present at the root of the filesystem, e.g. "/bin" and "/lib".



ORGANIZING SOFTWARE

- Duplicate this directory structure for each version of a software package

path
/opt/shared/openmpi/1.8.3/bin
/opt/shared/openmpi/1.8.3/lib
/opt/shared/openmpi/1.8.3/share/man
/opt/shared/openmpi/1.8.3/include
/opt/shared/openmpi/1.8.3/lib/pkgconfig

Prefix for package:
/opt/shared/openmpi

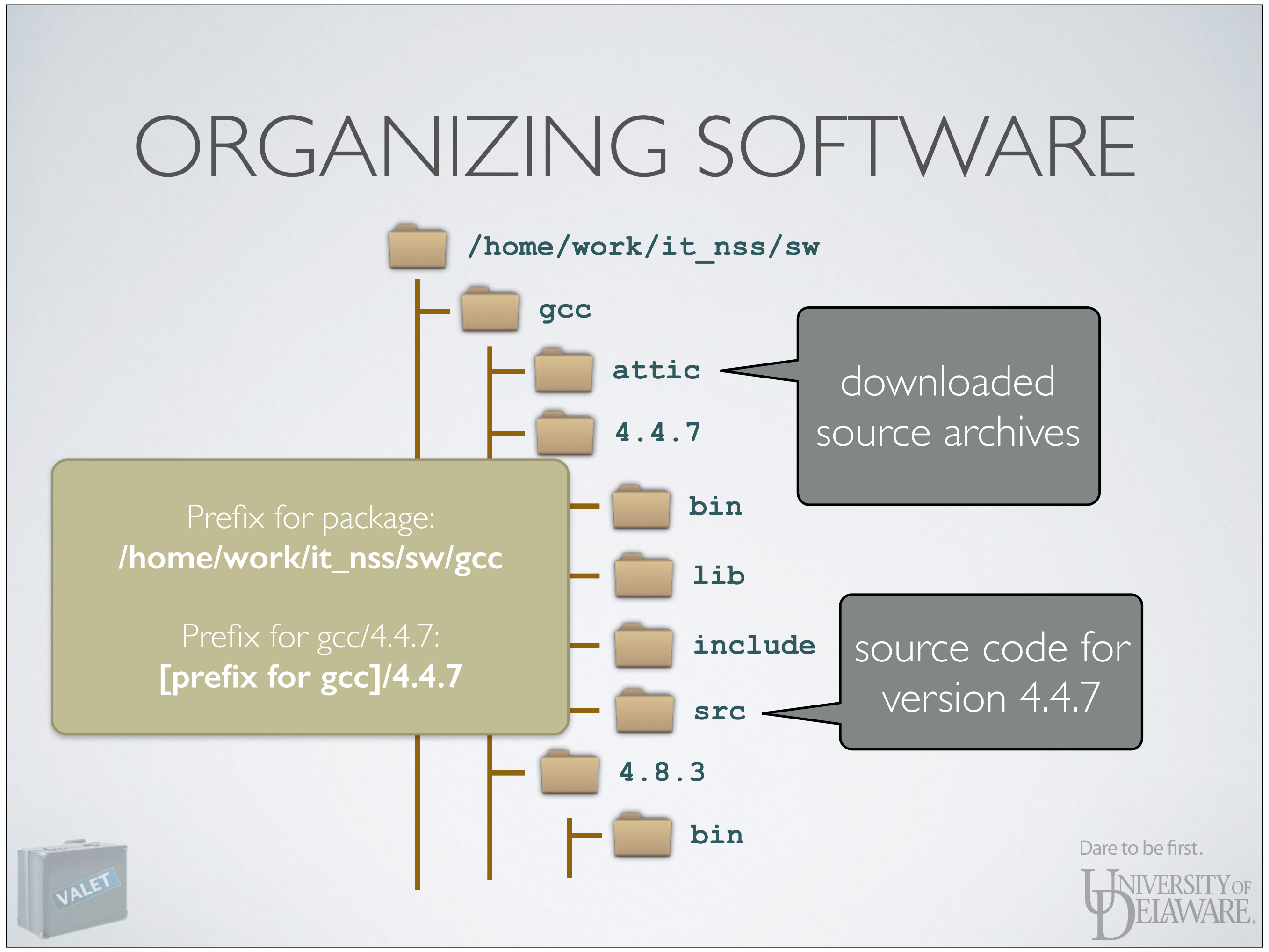
Prefix for version of package:
[prefix for package]/1.8.3

- Drop the "/usr" prefix and replace with a different prefix.
- The prefix for a version is relative to the prefix of the package.

ORGANIZING SOFTWARE

- Duplicate this directory structure for each version of a software package
  - Software built using the GNU ./configure system often install into this same set of directories
  - VALET looks for these paths' being present and will configure them accordingly
    - bin/ ➔ $PATH
    - lib64/ ➔ $LD_LIBRARY_PATH, $LDFLAGS
    - share/man ➔ $MANPATH
    - include/ ➔ $CPPFLAGS

Dare to be first.

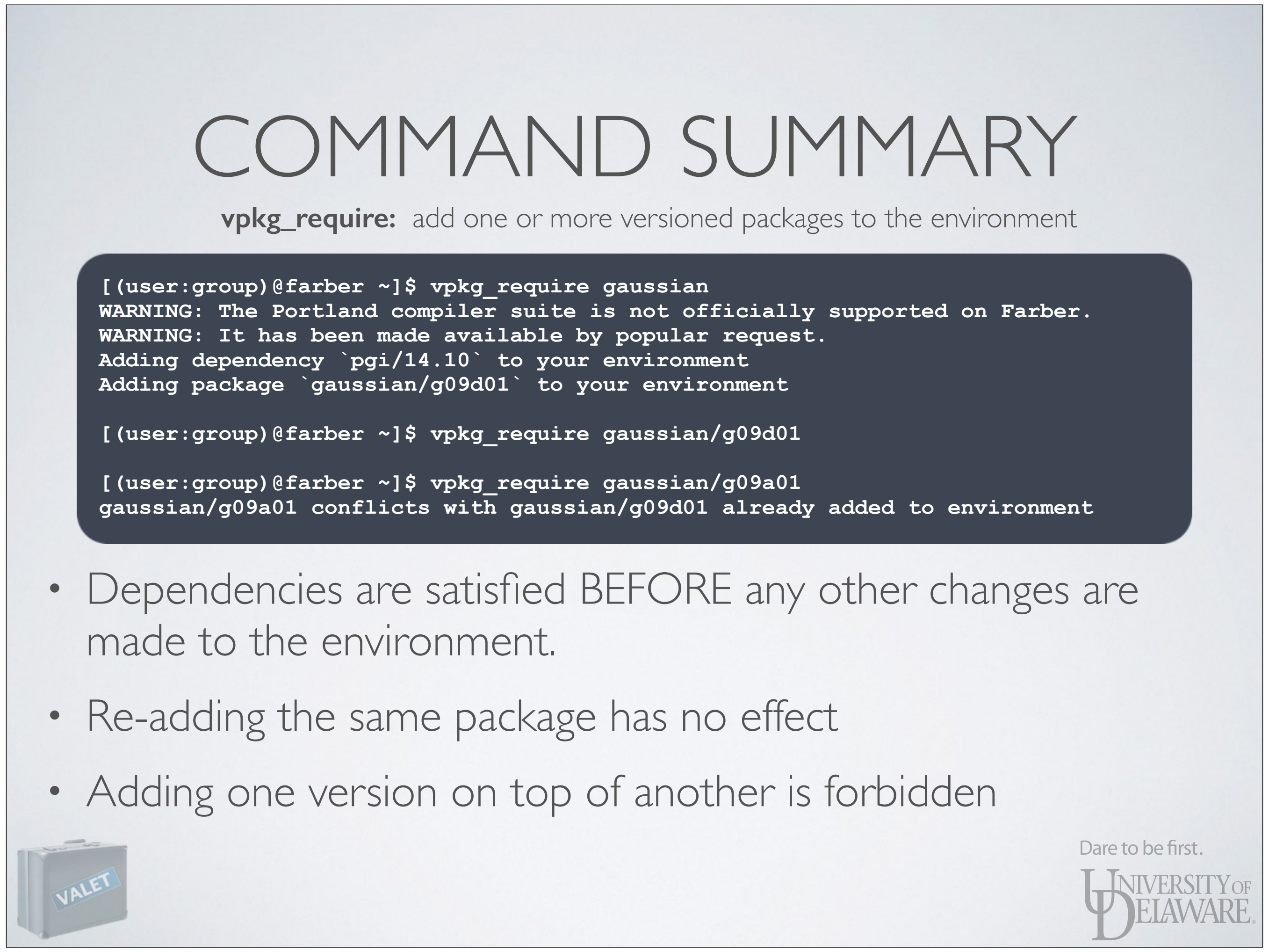


ORGANIZING SOFTWARE

Prefix for package:
/home/work/it_nss/sw/gcc

Prefix for gcc/4.4.7:
[prefix for gcc]/4.4.7

- This is the scheme that IT uses for the software it maintains on the clusters
- As mentioned a few slides back, adding a "valet" directory to this tree is an easy way to integrate with VALET.



COMMAND SUMMARY

vpkg_require: add one or more versioned packages to the environment

```
[(user:group)@farber ~]$ vpkg_require gaussian
WARNING: The Portland compiler suite is not officially supported on Farber.
WARNING: It has been made available by popular request.
Adding dependency `pgi/14.10` to your environment
Adding package `gaussian/g09d01` to your environment

[(user:group)@farber ~]$ vpkg_require gaussian/g09d01

[(user:group)@farber ~]$ vpkg_require gaussian/g09a01
gaussian/g09a01 conflicts with gaussian/g09d01 already added to environment
```

- Dependencies are satisfied BEFORE any other changes are made to the environment.
- Re-adding the same package has no effect
- Adding one version on top of another is forbidden

Dare to be first.

-

COMMAND SUMMARY

vpkg_rollback: remove environment changes introduced by **vpkg_require**

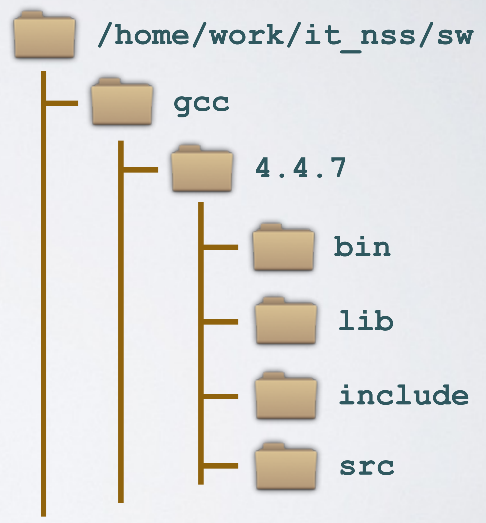
```
[(user:group)@farber ~]$ vpkg_rollback

[(user:group)@farber ~]$ vpkg_rollback
ERROR:  no previous session on record, unable to roll back

[(user:group)@farber ~]$ vpkg_rollback all
```

- Each **vpkg_require** creates a snapshot of the full environment prior to making any changes
- The **vpkg_rollback** command restores the last snapshot
  - Include the argument **all** to remove the effects of all preceding **vpkg_require**'s performed in the shell

Dare to be first.

WRITING VALET PACKAGE DEFINITIONS

- Package prefix
  `/home/work/it_nss/sw/gcc`
- Version prefix (relative)
  `4.4.7`
- Uses standard paths



WRITING VALET PACKAGE DEFINITIONS

- Please note:
  - If a prefix is not provided for a version, its id is assumed to be the relative prefix
  - Using standard paths, the configuration is extremely simple to express

```
<package id="gcc">
  <prefix>/home/work/it_nss/sw/gcc</prefix>
  <version id="4.4.7">
  </version>
</package>
```

WRITING VALET PACKAGE DEFINITIONS

- Same package definition expressed in JSON

```
{
  "gcc": {
    "prefix": "/home/work/it_nss/sw/gcc",
    "versions": {
      "4.4.7": {
      }
    }
  }
}
```

WRITING VALET PACKAGE DEFINITIONS

- Explicitly configure those standard paths
  - Indicate that standard paths should not be implicitly managed
  - Add *actions* to the version's configuration

```
<package id="gcc">
  <prefix>/home/work/it_nss/sw/gcc</prefix>
  <no-standard-paths/>
  <version id="4.4.7">
    <actions>
      <bindir>bin</bindir>
      <incdir>include</incdir>
      <libdir>lib</libdir>
    </actions>
  </version>
</package>
```

WRITING VALET PACKAGE DEFINITIONS

- Same package definition expressed in JSON

```
{
  "gcc": {
    "prefix": "/home/work/it_nss/sw/gcc",
    "standard-paths": false,
    "versions": {
      "4.4.7": {
        "actions": [
          {
            "bindir": "bin",
            "incdir": "include",
            "libdir": "lib"
          }
        ]
      }
    }
  }
}
```

WRITING VALET PACKAGE DEFINITIONS

- Create a version "4.4" that always equa newest 4.4
  - Add a *ver*
  - Use "4.4" version

without an explicit default, the **first** version specified is assumed to be the default for the package

```
<package id="gcc">
  <prefix>/home/work/it_nss/sw/gcc</prefix>
  <no-standard-paths/>
  <default-version>4.4</default-version>
  <version id="4.4.7">
    <actions>
      <bindir>bin</bindir>
      <incdir>include</incdir>
      <libdir>lib</libdir>
    </actions>
  </version>
  <version id="4.4" alias-to="4.4.7"/>
</package>
```

WRITING VALET PACKAGE DEFINITIONS

- Same package definition expressed in JSON

```
{
  "gcc": {
    "prefix": "/home/work/it_nss/sw/gcc",
    "standard-paths": false,
    "default-version": "4.4",
    "versions": {
      "4.4.7": {
        "actions": [
          {
            "bindir": "bin",
            "incdir": "include",
            "libdir": "lib"
          }
        ]
      },
      "4.4": {
        "alias-to": "4.4.7"
      }
    }
  }
}
```

WRITING VALET PACKAGE DEFINITIONS

- Add a command alias, configured with any version, which displays the version of GCC
  - An actions list is valid both for a version and for the package itself

```
<package id="gcc">
  <prefix>/home/work/it_nss/sw/gcc</prefix>
  <default-version>4.4</default-version>
  <actions>
    <shell-alias
      shell="sh"
      name="gcc_version">
         gcc -v 2&gt;&amp;1 | tail -1
    </shell-alias>
  </actions>
  <version id="4.4.7"></version>
  <version id="4.4" alias-to="4.4.7"/>
</package>
```

- We'll switch back to allowing VALET to recognize and add standard paths

WRITING VALET PACKAGE DEFINITIONS

- Same package definition expressed in JSON

```
{
  "gcc": {
    "prefix": "/home/work/it_nss/sw/gcc",
    "default-version": "4.4",
    "actions": [
      { "shell-alias": "gcc_version",
        "command": {
          "sh": "gcc -v 2>&1 | tail -1"
        }
      }
    ],
    "versions": {
      "4.4.7": {},
      "4.4": {
        "alias-to": "4.4.7"
      }
    }
  }
}
```

WRITING VALET PACKAGE DEFINITIONS

- Install GCC 4.8.3, which makes use of MPFR 3.1.2 and any 1.x version of MPC 1.0.2
  - Uses a regular expression for the MPC version id
    - ANY version starting with "1." is acceptable

```
<package id="gcc">
  <prefix>/home/work/it_nss/sw/gcc</prefix>
  <default-version>4.4</default-version>
  <actions>
    <shell-alias
      shell="sh"
      name="gcc_version">
         gcc -v 2&gt;&amp;1 | tail -1
    </shell-alias>
  </actions>
  <version id="4.8.3">
    <dependencies>
      <package id="mpfr/3.1.2"/>
      <package id="mpc/^^1\."/>
    </dependencies>
  </version>
  <version id="4.4.7"></version>
  <version id="4.4" alias-to="4.4.7"/>
</package>
```

WRITING VALET PACKAGE DEFINITIONS

- Same package definition expressed in JSON

```
{
  "gcc": {
    "prefix": "/home/work/it_nss/sw/gcc",
    "default-version": "4.4",
    "actions": [
      { "shell-alias": "gcc_version",
        "command": {
          "sh": "gcc -v 2>&1 | tail -1"
        }
      }
    ],
    "versions": {
      "4.8.3": {
        "dependencies": [
          "mpfr/3.0.2",
          "mpc/^^1\\."
        ]
      },
      "4.4.7": {},
      "4.4": {
        "alias-to": "4.4.7"
      }
    }
  }
}
```

WRITING VALET PACKAGE DEFINITIONS

- Check your VALET package definition for correctness
  - The **vpkg_check** command will attempt to parse a file and display errors if unsuccessful

```
[(user:group)@farber .valet]$ vpkg_check dummy.vpkg
ERROR:   dummy.vpkg is not a valid XML file: not well-formed (invalid token): line 7,
column 19

[(user:group)@farber .valet]$ vpkg_check dummy.vpkg_json
ERROR:   dummy.vpkg_json is not a valid JSON file: Invalid \escape: line 16 column 19
(char 331):: {
::   "dummy": {
::     "prefix": "/home/work/it_nss/sw/gcc",
::
```

Dare to be first.
University of Delaware

WRITING VALET PACKAGE DEFINITIONS

- Just the tip of the iceberg!
- Extensive documentation of the XML and JSON grammar can be found at

http://docs.hpc.udel.edu/software/valet/start

Dare to be first.

UNIVERSITY OF DELAWARE

SUMMARY

- Managing environment configuration is key to working safely and smartly
- Adopting a modular, organized approach to installing software helps promote that
- Automation via tools like environment modules or VALET saves a great deal of frustration, time, and effort

SUMMARY

- VALET provides a mechanism for modeling environment alterations associated with software packages…
  - …that can be very simple (our initial GCC example).
  - …or very complex when necessary.
- VALET uses full environment checkpointing for accurate reversion of changes to the environment

VALET

Dare to be first.

UNIVERSITY OF DELAWARE.

Any Questions?

http://docs.hpc.udel.edu/software/valet/start

Dare to be first.