SOFTWARE MANAGEMENT WITH VALET

Dr. Jeffrey Frey
University of Delaware, IT

Dare to be first.

UNIVERSITY OF DELAWARE

SOFTWARE INSTALLATION WORKFLOW

- Like any computational task, installing software has a workflow associated with it
    - Creating — and adhering to — a *standard workflow* for software installation makes *managing* software easier

SOFTWARE INSTALLATION WORKFLOW

- The previous seminar discussed the various methods for *building* software from *source code*
  - GNU autotools (e.g. `./configure`)
  - CMake

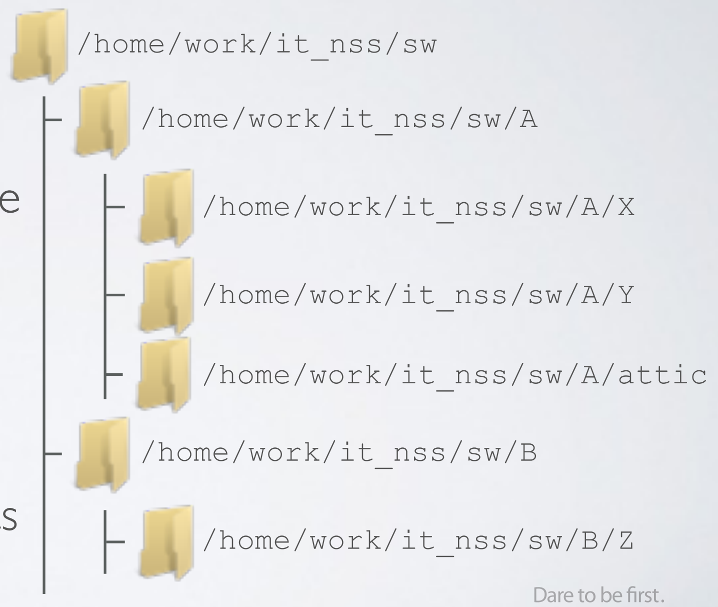
SOFTWARE INSTALLATION WORKFLOW

- The previous seminar discussed the various methods for *building* software from *source code*
- The seminar prior to that covered your options re: the choice of toolset for *building* the software
  - GCC, Intel, Portland, etc.

Dare to be first.

SOFTWARE INSTALLATION WORKFLOW

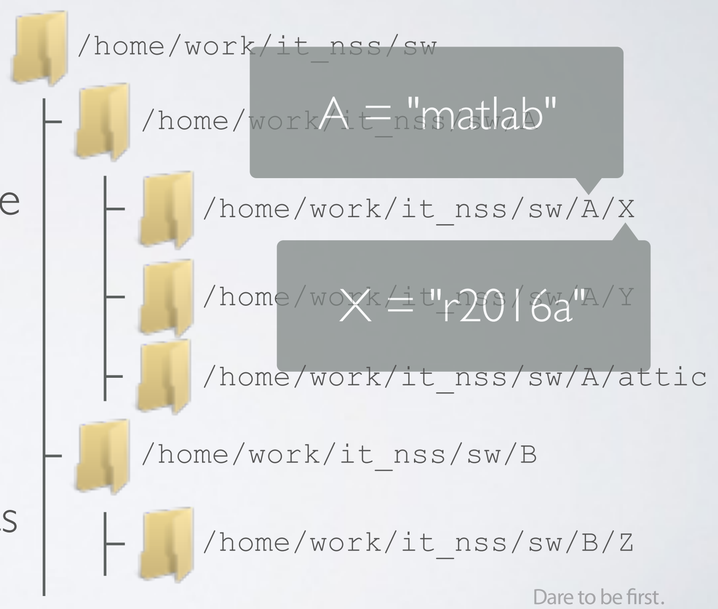
- The *standard workflow* endorsed on our clusters compartmentalizes each version of each software title
  - All software is installed in a common directory (e.g. `/home/work/it_nss/sw`)
  - Titles and versions exist as distinct directories



```
/home/work/it_nss/sw
├ /home/work/it_nss/sw/A
│  ├ /home/work/it_nss/sw/A/X
│  ├ /home/work/it_nss/sw/A/Y
│  └ /home/work/it_nss/sw/A/attic
└ /home/work/it_nss/sw/B
   └ /home/work/it_nss/sw/B/Z
```

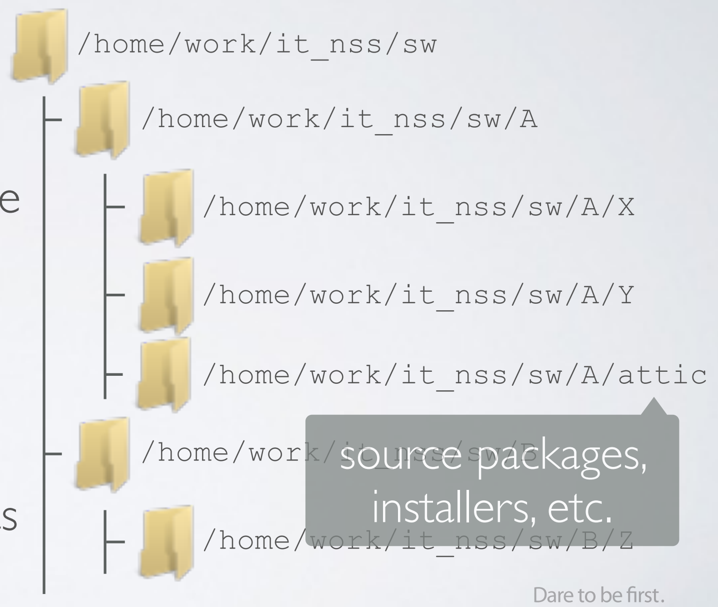
SOFTWARE INSTALLATION WORKFLOW

- The *standard workflow* endorsed on our clusters compartmentalizes each version of each software title
  - All software is installed in a common directory (e.g. `/home/work/it_nss/sw`)
  - Titles and versions exist as distinct directories



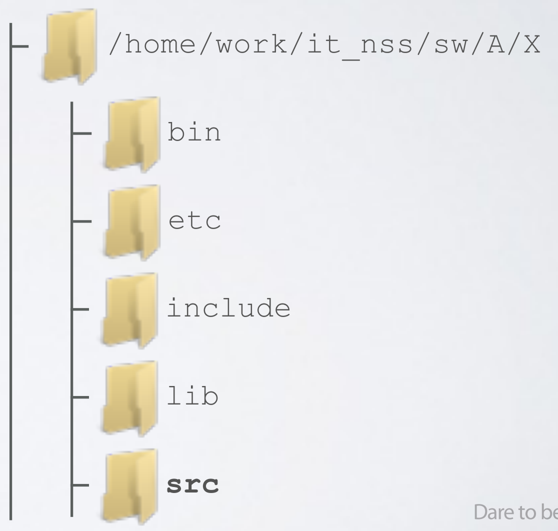
SOFTWARE INSTALLATION WORKFLOW

- The *standard workflow* endorsed on our clusters compartmentalizes each version of each software title
  - All software is installed in a common directory (e.g. `/home/work/it_nss/sw`)
  - Titles and versions exist as distinct directories



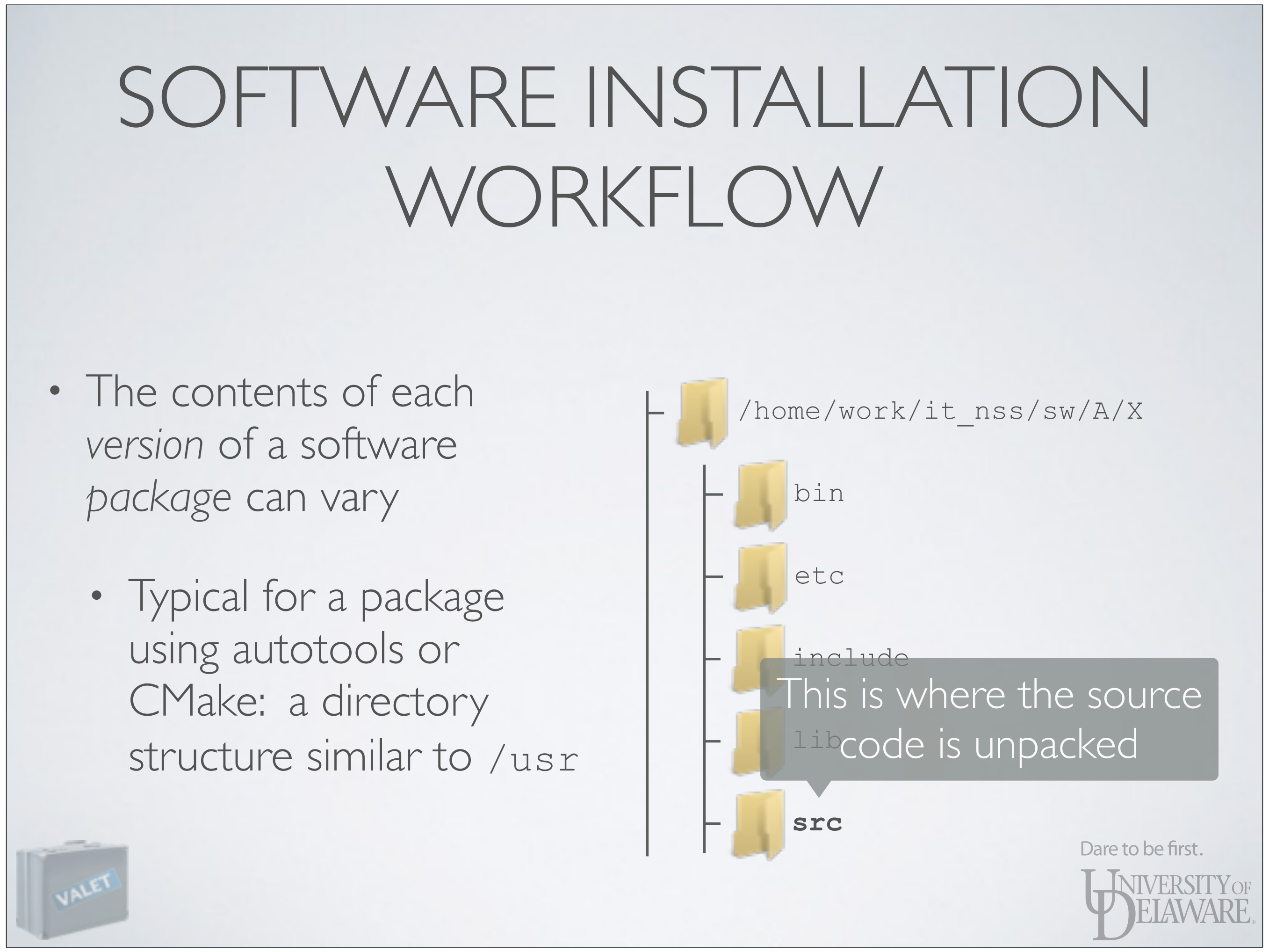
SOFTWARE INSTALLATION WORKFLOW

- The contents of each *version* of a software *package* can vary
  - Typical for a package using autotools or CMake: a directory structure similar to `/usr`



Dare to be first.

Demonstration: unpack step7-1.0 source



SOFTWARE INSTALLATION WORKFLOW

- The contents of each *version* of a software *package* can vary
  - Typical for a package using autotools or CMake: a directory structure similar to `/usr`

```
/home/work/it_nss/sw/A/X
    bin
    etc
    include
    lib
    src
```

This is where the source code is unpacked

Demonstration: unpack step7-1.0 source

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber step7]$ pwd
/home/work/it_nss/sw/step7

[frey@farber step7]$ ls
total 4
drwxr-sr-x 2 frey it_nss 4 Apr 15 12:39 attic

[frey@farber step7]$ ls attic
total 9
-rw-r--r-- 1 frey it_nss 2546 Apr 15 12:36 tutorial-1.0.tar.gz
-rw-r--r-- 1 frey it_nss 2590 Apr 15 12:51 tutorial-1.1.tar.gz
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber step7]$ mkdir 1.0

[frey@farber step7]$ cd 1.0

[frey@farber 1.0]$ tar -zxf ../attic/tutorial-1.0.tar.gz

[frey@farber 1.0]$ mv tutorial-1.0 src

[frey@farber 1.0]$ cd src

[frey@farber src]$ pwd
/home/work/it_nss/sw/step7/1.0/src

[frey@farber src]$ ls
total 15
-rw-r--r-- 1 frey it_nss 2326 Jul 23  2015 CMakeLists.txt
-rw-r--r-- 1 frey it_nss   85 Jul 23  2015 License.txt
drwxr-sr-x 2 frey it_nss    6 Apr 15 10:57 MathFunctions
-rw-r--r-- 1 frey it_nss  286 Jul 23  2015 TutorialConfig.h.in
-rw-r--r-- 1 frey it_nss  769 Jul 23  2015 tutorial.cxx
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber build]$ vpkg_require cmake/3.3
Adding package `cmake/3.3.0` to your environment

[frey@farber src]$ mkdir build
[frey@farber src]$ cd build

[frey@farber build]$ cmake -D CMAKE_INSTALL_PREFIX=/home/work/it_nss/sw/step7/1.0 ..
-- The C compiler identification is GNU 4.4.7
-- The CXX compiler identification is GNU 4.4.7
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Looking for log
-- Looking for log - not found
-- Looking for exp
-- Looking for exp - not found
-- Configuring done
-- Generating done
-- Build files have been written to: /home/work/it_nss/sw/step7/1.0/src/build
```

SOFTWARE INSTALLATION WORKFLOW

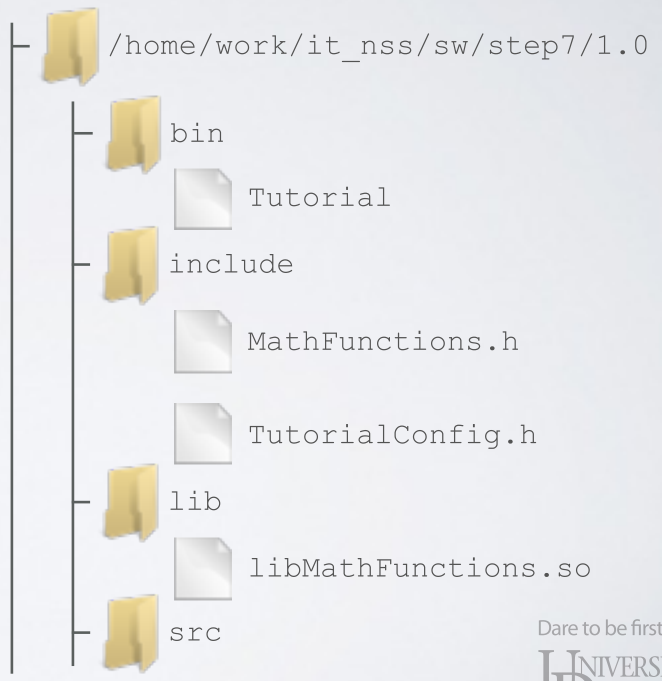
```
[frey@farber build]$ make
Scanning dependencies of target MakeTable
[ 14%] Building CXX object MathFunctions/CMakeFiles/MakeTable.dir/MakeTable.cxx.o
[ 28%] Linking CXX executable MakeTable
[ 28%] Built target MakeTable
[ 42%] Generating Table.h
Scanning dependencies of target MathFunctions
[ 57%] Building CXX object MathFunctions/CMakeFiles/MathFunctions.dir/mysqrt.cxx.o
[ 71%] Linking CXX shared library libMathFunctions.so
[ 71%] Built target MathFunctions
Scanning dependencies of target Tutorial
[ 85%] Building CXX object CMakeFiles/Tutorial.dir/tutorial.cxx.o
[100%] Linking CXX executable Tutorial
[100%] Built target Tutorial

[frey@farber build]$ make install
[ 28%] Built target MakeTable
[ 71%] Built target MathFunctions
[100%] Built target Tutorial
Install the project...
-- Install configuration: ""
-- Installing: /home/work/it_nss/sw/step7/1.0/bin/Tutorial
-- Removed runtime path from "/home/work/it_nss/sw/step7/1.0/bin/Tutorial"
-- Installing: /home/work/it_nss/sw/step7/1.0/include/TutorialConfig.h
-- Installing: /home/work/it_nss/sw/step7/1.0/lib/libMathFunctions.so
-- Up-to-date: /home/work/it_nss/sw/step7/1.0/include/MathFunctions.h
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber 1.0]$ pwd
/home/work/it_nss/sw/step7/1.0

[frey@farber 1.0]$ ls -l
total 14
drwxr-sr-x 2 frey it_nss 3 Apr 18 14:27 bin
drwxr-sr-x 2 frey it_nss 4 Apr 18 14:27 include
drwxr-sr-x 2 frey it_nss 3 Apr 18 14:27 lib
drwxr-sr-x 4 frey it_nss 8 Apr 18 14:24 src

[frey@farber 1.0]$ ls -l bin
total 7
-rwxr-xr-x 1 frey it_nss 7963 Apr 18 14:26 Tutorial

[frey@farber 1.0]$ ls -l include
total 4
-rw-r--r-- 1 frey it_nss  25 Jul 23  2015 MathFunctions.h
-rw-r--r-- 1 frey it_nss 235 Apr 18 14:25 TutorialConfig.h

[frey@farber 1.0]$ ls -l lib
total 5
-rwxr-xr-x 1 frey it_nss 6963 Apr 18 14:26 libMathFunctions.so
```

SOFTWARE INSTALLATION WORKFLOW

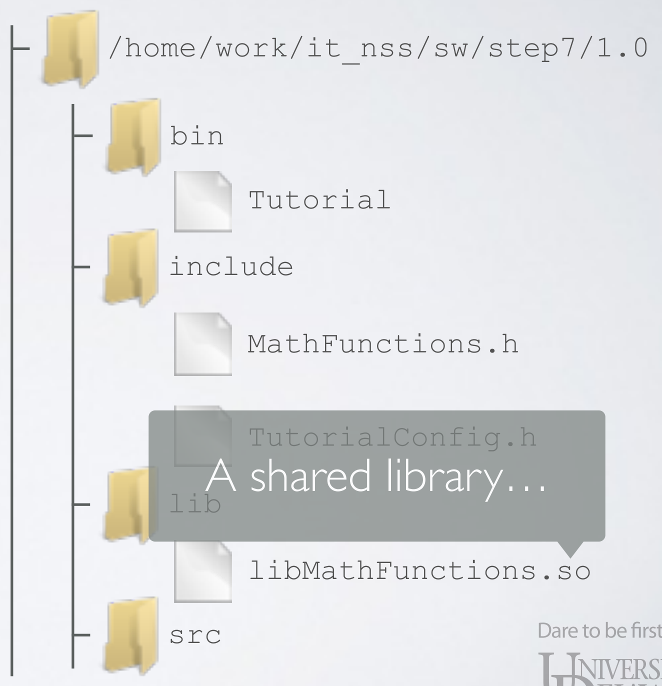
- In pictorial fashion:

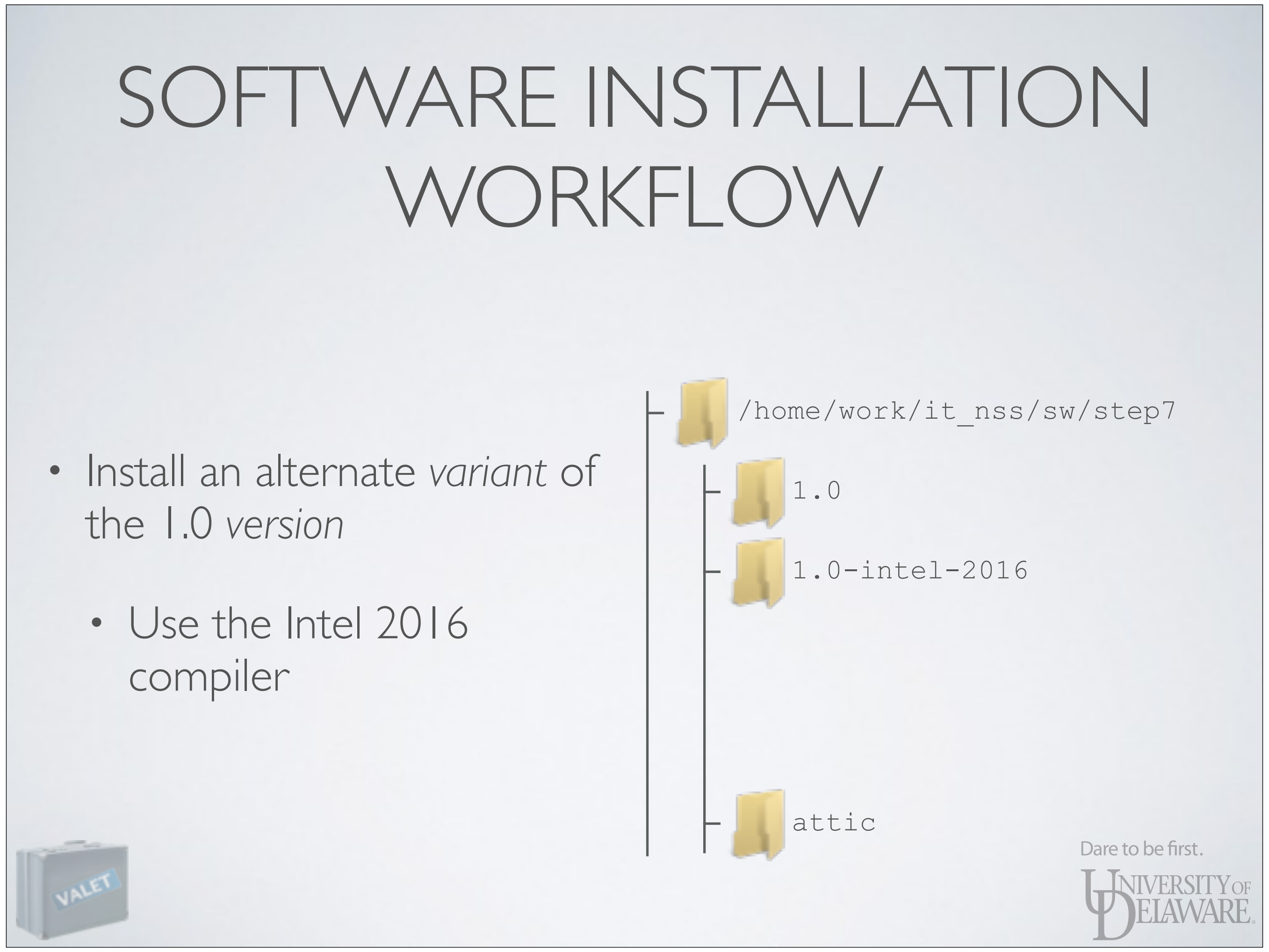


```
/home/work/it_nss/sw/step7/1.0
    bin
        Tutorial
    include
        MathFunctions.h
        TutorialConfig.h
    lib
        libMathFunctions.so
    src
```

SOFTWARE INSTALLATION WORKFLOW

- In pictorial fashion:





SOFTWARE INSTALLATION WORKFLOW

- Install an alternate *variant* of the 1.0 *version*
    - Use the Intel 2016 compiler

```
├ /home/work/it_nss/sw/step7
  ├ 1.0
  ├ 1.0-intel-2016
  ├ attic
```

Dare to be first.

UNIVERSITY OF DELAWARE

Demonstration: build with alternate compiler

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber step7]$ mkdir 1.0-intel-2016

[frey@farber step7]$ cd 1.0-intel-2016

[frey@farber 1.0-intel-2016]$ tar -zxf ../attic/tutorial-1.0.tar.gz

[frey@farber 1.0-intel-2016]$ mv tutorial-1.0 src

[frey@farber 1.0-intel-2016]$ cd src

[frey@farber src]$ pwd
/home/work/it_nss/sw/step7/1.0-intel-2016/src

[frey@farber src]$ ls
total 15
-rw-r--r-- 1 frey it_nss 2326 Jul 23  2015 CMakeLists.txt
-rw-r--r-- 1 frey it_nss   85 Jul 23  2015 License.txt
drwxr-sr-x 2 frey it_nss    6 Apr 15 10:57 MathFunctions
-rw-r--r-- 1 frey it_nss  286 Jul 23  2015 TutorialConfig.h.in
-rw-r--r-- 1 frey it_nss  769 Jul 23  2015 tutorial.cxx
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber build]$ vpkg_require cmake/3.3 intel/2016
Adding package `intel/2016.2.062` to your environment

[frey@farber src]$ mkdir build
[frey@farber src]$ cd build

[frey@farber build]$ export CC=icc; export CXX=icpc
[frey@farber build]$ cmake -D CMAKE_INSTALL_PREFIX=/home/work/it_nss/sw/step7/1.0-intel-2016 ..
-- The C compiler identification is Intel 16.0.2.20160204
-- The CXX compiler identification is Intel 16.0.2.20160204
-- Check for working C compiler: /opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/bin/intel64/icc
-- Check for working C compiler: /opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/bin/intel64/icc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/bin/intel64/icpc
-- Check for working CXX compiler: /opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/bin/intel64/icpc -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Looking for log
-- Looking for log - found
-- Looking for exp
-- Looking for exp - found
-- Configuring done
-- Generating done
-- Build files have been written to: /home/work/it_nss/sw/step7/1.0-intel-2016/src/build
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber build]$ make
Scanning dependencies of target MakeTable
[ 14%] Building CXX object MathFunctions/CMakeFiles/MakeTable.dir/MakeTable.cxx.o
[ 28%] Linking CXX executable MakeTable
[ 28%] Built target MakeTable
[ 42%] Generating Table.h
Scanning dependencies of target MathFunctions
[ 57%] Building CXX object MathFunctions/CMakeFiles/MathFunctions.dir/mysqrt.cxx.o
[ 71%] Linking CXX shared library libMathFunctions.so
[ 71%] Built target MathFunctions
Scanning dependencies of target Tutorial
[ 85%] Building CXX object CMakeFiles/Tutorial.dir/tutorial.cxx.o
[100%] Linking CXX executable Tutorial
[100%] Built target Tutorial

[frey@farber build]$ make install
[ 28%] Built target MakeTable
[ 71%] Built target MathFunctions
[100%] Built target Tutorial
Install the project...
-- Install configuration: ""
-- Installing: /home/work/it_nss/sw/step7/1.0-intel-2016/bin/Tutorial
-- Removed runtime path from "/home/work/it_nss/sw/step7/1.0-intel-2016/bin/Tutorial"
-- Installing: /home/work/it_nss/sw/step7/1.0-intel-2016/include/TutorialConfig.h
-- Installing: /home/work/it_nss/sw/step7/1.0-intel-2016/lib/libMathFunctions.so
-- Installing: /home/work/it_nss/sw/step7/1.0-intel-2016/include/MathFunctions.h
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber 1.0-intel-2016]$ pwd
/home/work/it_nss/sw/step7/1.0-intel-2016

[frey@farber 1.0-intel-2016]$ ls -l
total 14
drwxr-sr-x 2 frey it_nss 3 Apr 18 14:39 bin
drwxr-sr-x 2 frey it_nss 4 Apr 18 14:39 include
drwxr-sr-x 2 frey it_nss 3 Apr 18 14:39 lib
drwxr-sr-x 4 frey it_nss 8 Apr 18 14:37 src

[frey@farber 1.0-intel-2016]$ ls -l bin
total 14
-rwxr-xr-x 1 frey it_nss 21073 Apr 18 14:39 Tutorial

[frey@farber 1.0-intel-2016]$ ls -l include
total 4
-rw-r--r-- 1 frey it_nss  25 Jul 23  2015 MathFunctions.h
-rw-r--r-- 1 frey it_nss 225 Apr 18 14:38 TutorialConfig.h

[frey@farber 1.0-intel-2016]$ ls -l lib
total 5
-rwxr-xr-x 1 frey it_nss 8019 Apr 18 14:39 libMathFunctions.so
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber 1.0-intel-2016]$ bin/Tutorial
bin/Tutorial: error while loading shared libraries: libMathFunctions.so: cannot open shared
object file: No such file or directory

[frey@farber 1.0-intel-2016]$ ls lib
total 5
-rwxr-xr-x 1 frey it_nss 8019 Apr 18 14:39 libMathFunctions.so
```

SOFTWARE INSTALLATION WORKFLOW

```
[frey@farber 1.0-intel-2016]$ bin/Tutorial
bin/Tutorial: error while loading shared libraries: libMathFunctions.so: cannot open shared
object file: No such file or directory

[frey@farber 1.0-intel-2016]$ ls lib
total 5
-rwxr-xr-x 1 frey it_nss 8019 Apr 18 14:39 libMathFunctions.so

[frey@farber 1.0-intel-2016]$ export LD_LIBRARY_PATH="/home/work/it_nss/sw/step7/1.0-intel-2016/
lib:${LD_LIBRARY_PATH}"

[frey@farber 1.0-intel-2016]$ bin/Tutorial
bin/Tutorial Version 1.0
Usage: bin/Tutorial number
```

SOFTWARE INSTALLATION WORKFLOW

```
frey@farber 1.0-intel-2016]$ echo $LD_LIBRARY_PATH
/home/work/it_nss/sw/step7/1.0-intel-2016/lib:/opt/intel/mic/coi/host-linux-release/lib:/opt/
intel/mic/myo/lib:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mkl/
lib/intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/compiler/lib/
intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/compiler/lib/
intel64_lin:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mpi/intel64/
lib:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mpi/mic/lib:/opt/
shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/ipp/lib/intel64:/opt/intel/
mic/coi/host-linux-release/lib:/opt/intel/mic/myo/lib:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/compiler/lib/intel64:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/mkl/lib/intel64:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/tbb/lib/intel64/gcc4.4:/opt/shared/intel/2016-Update2/
debugger_2016/libipt/intel64/lib:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/daal/lib/intel64_lin:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/daal/../tbb/lib/intel64_lin/gcc4.4:/opt/shared/intel/
2016-Update2/compilers_and_libraries_2016.2.181/linux/daal/../compiler/lib/intel64_lin
```

```
[frey@farber build]$ vpkg_require cmake/3.3 intel/2016
Adding package `intel/2016.2.062` to your environment
```

SOFTWARE INSTALLATION WORKFLOW

```
frey@farber 1.0-intel-2016]$ echo $LD_LIBRARY_PATH
/home/work/it_nss/sw/step7/1.0-intel-2016/lib:/opt/intel/mic/coi/host-linux-release/lib:/opt/
intel/mic/myo/lib:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mkl/
lib/intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/compiler/lib/
intel64:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/compiler/lib/
intel64_lin:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mpi/intel64/
lib:/opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/mpi/mic/lib:/opt/
shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/ipp/lib/intel64:/opt/intel/
mic/coi/host-linux-release/lib:/opt/intel/mic/myo/lib:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/compiler/lib/intel64:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/mkl/lib/intel64:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/tbb/lib/intel64/gcc4.4:/opt/shared/intel/2016-Update2/
debugger_2016/libipt/intel64/lib:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/daal/lib/intel64_lin:/opt/shared/intel/2016-Update2/
compilers_and_libraries_2016.2.181/linux/daal/../tbb/lib/intel64_lin/gcc4.4:/opt/shared/intel/
2016-Update2/compilers_and_libraries_2016.2.181/linux/daal/../compiler/lib/intel64_lin
```

Where did all THIS come from???

```
[frey@farber build]$ vpkg_require cmake/3.3 intel/2016
Adding package `intel/2016.2.062` to your environment
```

ENTER VALET…

- The **`vpkg_require`** command is used to configure a software package into the user's environment.
    - A *package definition file* provides VALET with a list of changes it must make
    - In this case, in part that means adding many paths to LD_LIBRARY_PATH

Dare to be first.
University of Delaware

CREATING PACKAGE DEFINITION FILES

- On Farber, VALET understands an XML file format and a JSON file format
  - JSON has a rigid syntax, but is easier to read
  - XML often looks "cluttered" due to markup

Dare to be first.
University of Delaware

CREATING PACKAGE DEFINITION FILES

- On Farber, VALET understands an XML file format and a JSON file format
- We will create a JSON-format *package definition file* for our "step7" software package

Dare to be first.

CREATING PACKAGE DEFINITION FILES

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":           "/home/work/it_nss/sw/step7",
    "description":      "CMake Tutorial, Step 7",
    "versions": {
      "1.0": {
        "description": "Version 1.0 (GCC compiler)",
        "prefix":      "/home/work/it_nss/sw/step7/1.0"
      },
      "1.0-intel-2016": {
        "description": "Version 1.0 (Intel 2016 compiler)",
        "prefix":      "/home/work/it_nss/sw/step7/1.0-intel-2016",
        "dependencies": [
          "intel/2016"
        ]
      }
    }
  }
}
```

CREATING PACKAGE DEFINITION FILES

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":         "/home/work/it_nss/sw/step7",
    "description":    "CMake Tutorial, Step 7",
    "versions": {
      "1.0": {
        "description": "Version 1.0 (GCC compiler)",
        "prefix":      "/home/work/it_nss/sw/step7/1.0"
      },
      "1.0-intel-2016": {
        "description": "Version 1.0 (Intel 2016 compiler)",
        "prefix":      "/home/work/it_nss/sw/step7/1.0-intel-2016",
        "dependencies": [
          "intel/2016"
        ]
      }
    }
  }
}
```

Version prefixes can be relative to the package prefix

CREATING PACKAGE DEFINITION FILES

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":           "/home/work/it_nss/sw/step7",
    "description":      "CMake Tutorial, Step 7",
    "versions": {
      "1.0": {
        "description": "Version 1.0 (GCC compiler)",
        "prefix":      "1.0"
      },
      "1.0-intel-2016": {
        "description": "Version 1.0 (Intel 2016 compiler)",
        "prefix":      "1.0-intel-2016",
        "dependencies": [
          "intel/2016"
        ]
      }
    }
  }
}
```

CREATING PACKAGE DEFINITION FILES

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":           "/home/work/it_nss/sw/step7",
    "description":      "CMake Tutorial, Step 7",
    "versions": {
      "1.0": {
        "description": "Version 1.0 (GCC compiler)",
        "prefix":      "1.0"
      },
      "1.0-intel-2016": {
        "description": "Version 1.0 (Intel 2016 compiler)",
        "prefix":      "1.0-intel-2016",
        "dependencies": [
          "intel/2016"
        ]
      }
    }
  }
}
```

If the version prefix = the version identifier, the prefix can be omitted

CREATING PACKAGE DEFINITION FILES

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":           "/home/work/it_nss/sw/step7",
    "description":      "CMake Tutorial, Step 7",
    "versions": {
      "1.0": {
        "description": "Version 1.0 (GCC compiler)"
      },
      "1.0-intel-2016": {
        "description": "Version 1.0 (Intel 2016 compiler)"
        "dependencies": [
          "intel/2016"
        ]
      }
    }
  }
}
```

CREATING PACKAGE DEFINITION FILES

- Where should *package definition files* be installed?

Path	Description
`~/.valet`	Personal software packages installed in e.g. `${HOME}`, `${WORKDIR}/user/X`
`${WORKDIR}/sw/valet`	Software for a workgroup installed in e.g. `${WORKDIR}/sw`

Dare to be first.
University of Delaware

CR
D

VALET automatically searches for packages in `~/.valet` and (after using the `workgroup` command) in `${WORKDIR}/sw/valet`.

It will also search any path you add to the `VALET_PATH` environment variable.

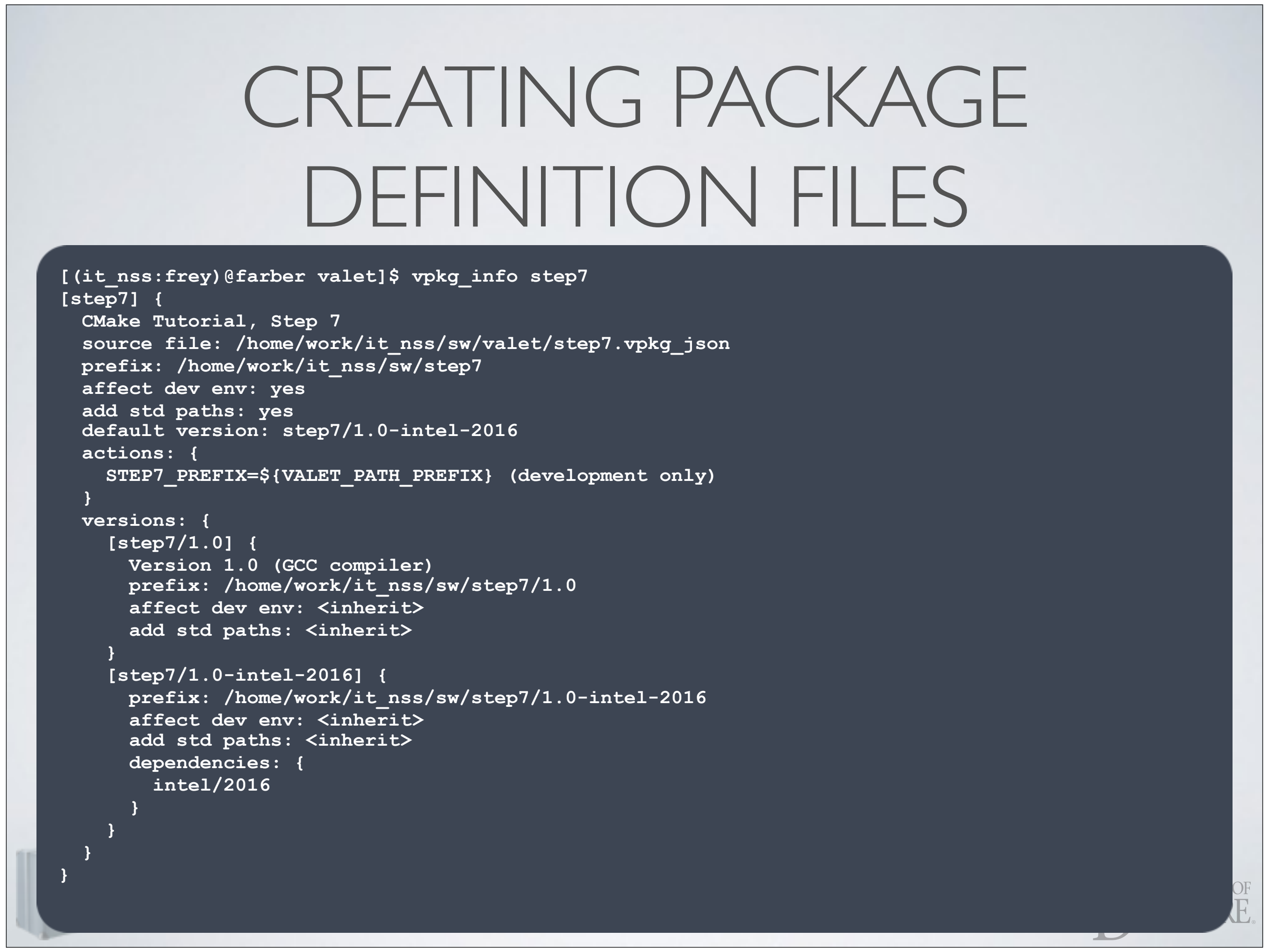
- Where shou

Path	Description
`~/.valet`	Personal software packages installed in e.g. `${HOME}`, `${WORKDIR}/user/X`
`${WORKDIR}/sw/valet`	Software for a workgroup installed in e.g. `${WORKDIR}/sw`

Dare to be first.
UNIVERSITY OF DELAWARE

CREATING PACKAGE DEFINITION FILES

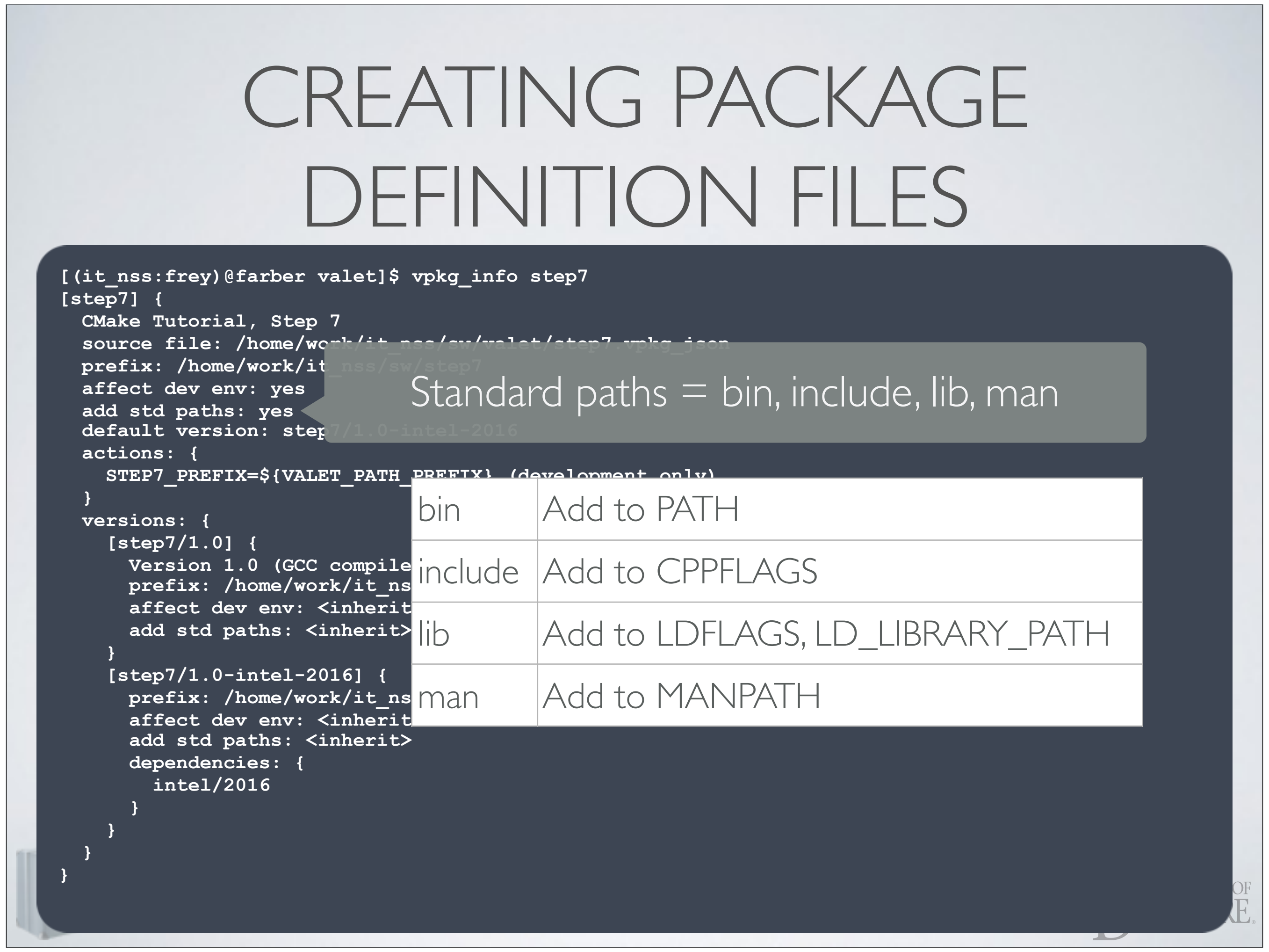
```
[frey@farber 1.0-intel-2016]$ workgroup -g it_nss

[(it_nss:frey)@farber 1.0-intel-2016]$ cd ${WORKDIR}/sw/valet

[(it_nss:frey)@farber valet]$ vi step7.vpkg_json
    :

[(it_nss:frey)@farber valet]$ cat step7.vpkg_json
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":      "/home/work/it_nss/sw/step7",
    "description": "CMake Tutorial, Step 7",
    "versions": {
       "1.0": {
         "description": "Version 1.0 (GCC compiler)"
       },
       "1.0-intel-2016": {
         "dependencies": [
           "intel/2016"
         ]
       }
    }
  }
}
```



CREATING PACKAGE DEFINITION FILES

```
[(it_nss:frey)@farber valet]$ vpkg_info step7
[step7] {
  CMake Tutorial, Step 7
  source file: /home/work/it_nss/sw/valet/step7.vpkg_json
  prefix: /home/work/it_nss/sw/step7
  affect dev env: yes
  add std paths: yes
  default version: step7/1.0-intel-2016
  actions: {
    STEP7_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [step7/1.0] {
      Version 1.0 (GCC compiler)
      prefix: /home/work/it_nss/sw/step7/1.0
      affect dev env: <inherit>
      add std paths: <inherit>
    }
    [step7/1.0-intel-2016] {
      prefix: /home/work/it_nss/sw/step7/1.0-intel-2016
      affect dev env: <inherit>
      add std paths: <inherit>
      dependencies: {
        intel/2016
      }
    }
  }
}
```

• With standard paths enabled, VALET looks inside the version's prefix path for a "bin" directory to add to the PATH, a "lib" directory to add to LD_LIBRARY_PATH, etc.



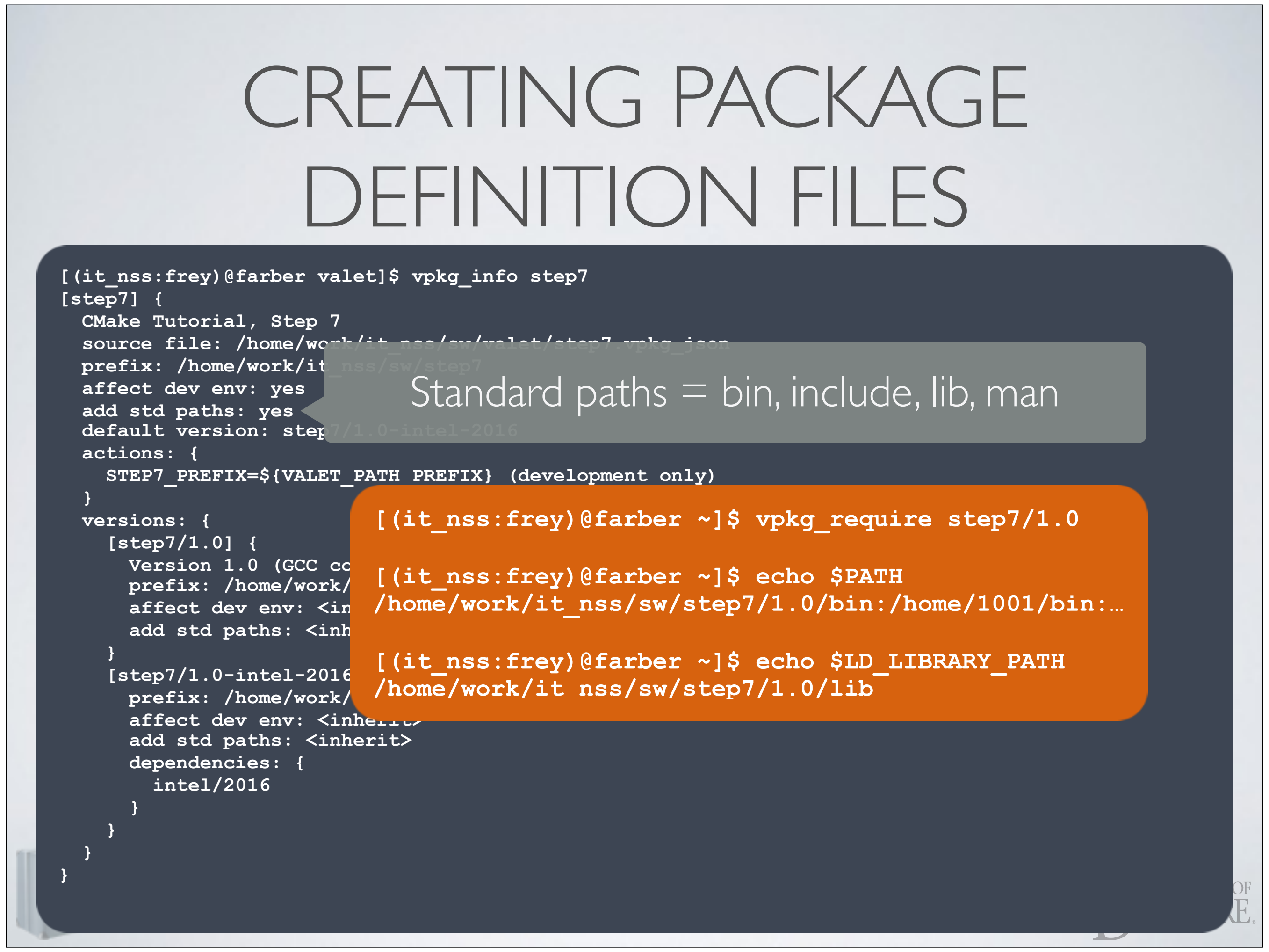
CREATING PACKAGE DEFINITION FILES

```
[(it_nss:frey)@farber valet]$ vpkg_info step7
[step7] {
  CMake Tutorial, Step 7
  source file: /home/work/it_nss/sw/valet/step7.vpkg_json
  prefix: /home/work/it_nss/sw/step7
  affect dev env: yes
  add std paths: yes
  default version: step7/1.0-intel-2016
  actions: {
    STEP7_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [step7/1.0] {
      Version 1.0 (GCC compile
      prefix: /home/work/it_ns
      affect dev env: <inherit
      add std paths: <inherit>
    }
    [step7/1.0-intel-2016] {
      prefix: /home/work/it_ns
      affect dev env: <inherit
      add std paths: <inherit>
      dependencies: {
        intel/2016
      }
    }
  }
}
```

Standard paths = bin, include, lib, man

bin	Add to PATH
include	Add to CPPFLAGS
lib	Add to LDFLAGS, LD_LIBRARY_PATH
man	Add to MANPATH

• With standard paths enabled, VALET looks inside the version's prefix path for a "bin" directory to add to the PATH, a "lib" directory to add to LD_LIBRARY_PATH, etc.



CREATING PACKAGE DEFINITION FILES

```
[(it_nss:frey)@farber valet]$ vpkg_info step7
[step7] {
  CMake Tutorial, Step 7
  source file: /home/work/it_nss/sw/valet/step7.vpkg_json
  prefix: /home/work/it_nss/sw/step7
  affect dev env: yes
  add std paths: yes
  default version: step7/1.0-intel-2016
  actions: {
    STEP7_PREFIX=${VALET_PATH PREFIX} (development only)
  }
  versions: {
    [step7/1.0] {
      Version 1.0 (GCC co
      prefix: /home/work/
      affect dev env: <in
      add std paths: <inh
    }
    [step7/1.0-intel-2016
      prefix: /home/work/
      affect dev env: <inherit>
      add std paths: <inherit>
      dependencies: {
        intel/2016
      }
    }
  }
}
```

Standard paths = bin, include, lib, man

```
[(it_nss:frey)@farber ~]$ vpkg_require step7/1.0

[(it_nss:frey)@farber ~]$ echo $PATH
/home/work/it_nss/sw/step7/1.0/bin:/home/1001/bin:…

[(it_nss:frey)@farber ~]$ echo $LD_LIBRARY_PATH
/home/work/it nss/sw/step7/1.0/lib
```

• With standard paths enabled, VALET looks inside the version's prefix path for a "bin" directory to add to the PATH, a "lib" directory to add to LD_LIBRARY_PATH, etc.

CREATING PACKAGE DEFINITION FILES

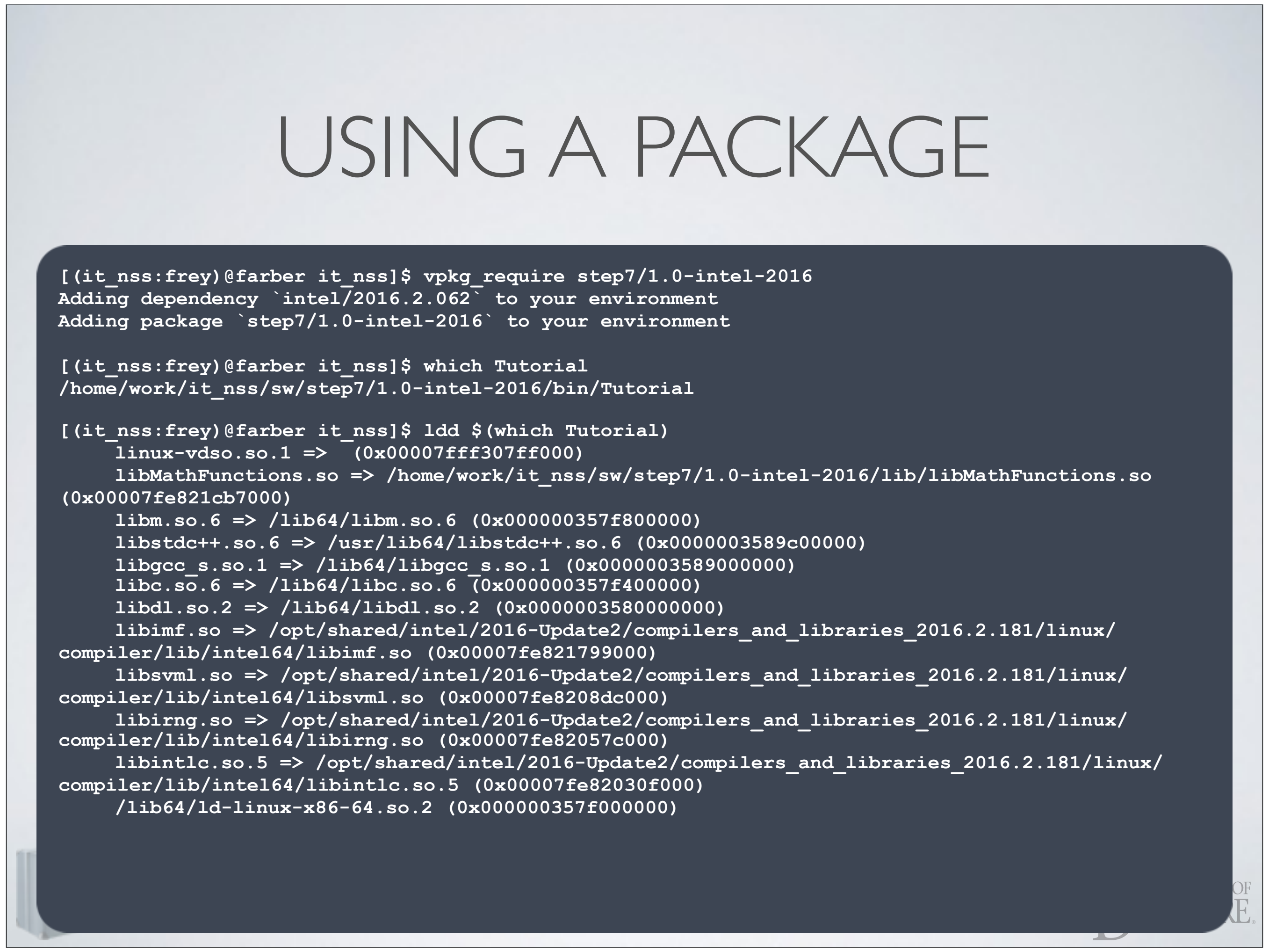
```
[(it_nss:frey)@farber valet]$ vpkg_info step7
[step7] {
  CMake Tutorial, Step 7
  source file: /home/work/it_nss/sw/valet/step7.vpkg_json
  prefix: /home/work/it_nss/sw/step7
  affect dev env: yes
  add std paths: yes
  default version: step7/1.0-intel-2016
  actions: {
    STEP7_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [step7/1.0] {
      Version 1.0 (GCC compiler)
      prefix: /home/work/it_nss/sw/step7/1.0
      affect dev env: <inherit>
      add std paths: <inherit>
    }
    [step7/1.0-intel-2016] {
      prefix: /home/work/it_nss/sw/step7/1.0-intel-2016
      affect dev env: <inherit>
      add std paths: <inherit>
      dependencies: {
        intel/2016
      }
    }
  }
}
```

CREATING PACKAGE DEFINITION FILES

```
[(it_nss:frey)@farber valet]$ vpkg_info step7
[step7] {
  CMake Tutorial, Step 7
  source file: /home/work/it_nss/sw/
  prefix: /home/work/it_nss/sw/st
  affect dev env: yes
  add std paths: yes
  default version: step7/1.0-intel-2016
  actions: {
    STEP7_PREFIX=${VALET_P
  }
  versions: {
    [step7/1.0] {
      Version 1.0 (GCC com
      prefix: /home/work/i
      affect dev env: <inh
      add std paths: <inhe
    }
    [step7/1.0-intel-2016] {
      prefix: /home/work/it_nss/sw/step7/1.0-intel-2016
      affect dev env: <inherit>
      add std paths: <inherit>
      dependencies: {
        intel/2016
      }
    }
  }
}
```

Using vpkg_devrequire will add CPPFLAGS and LDFLAGS to the environment

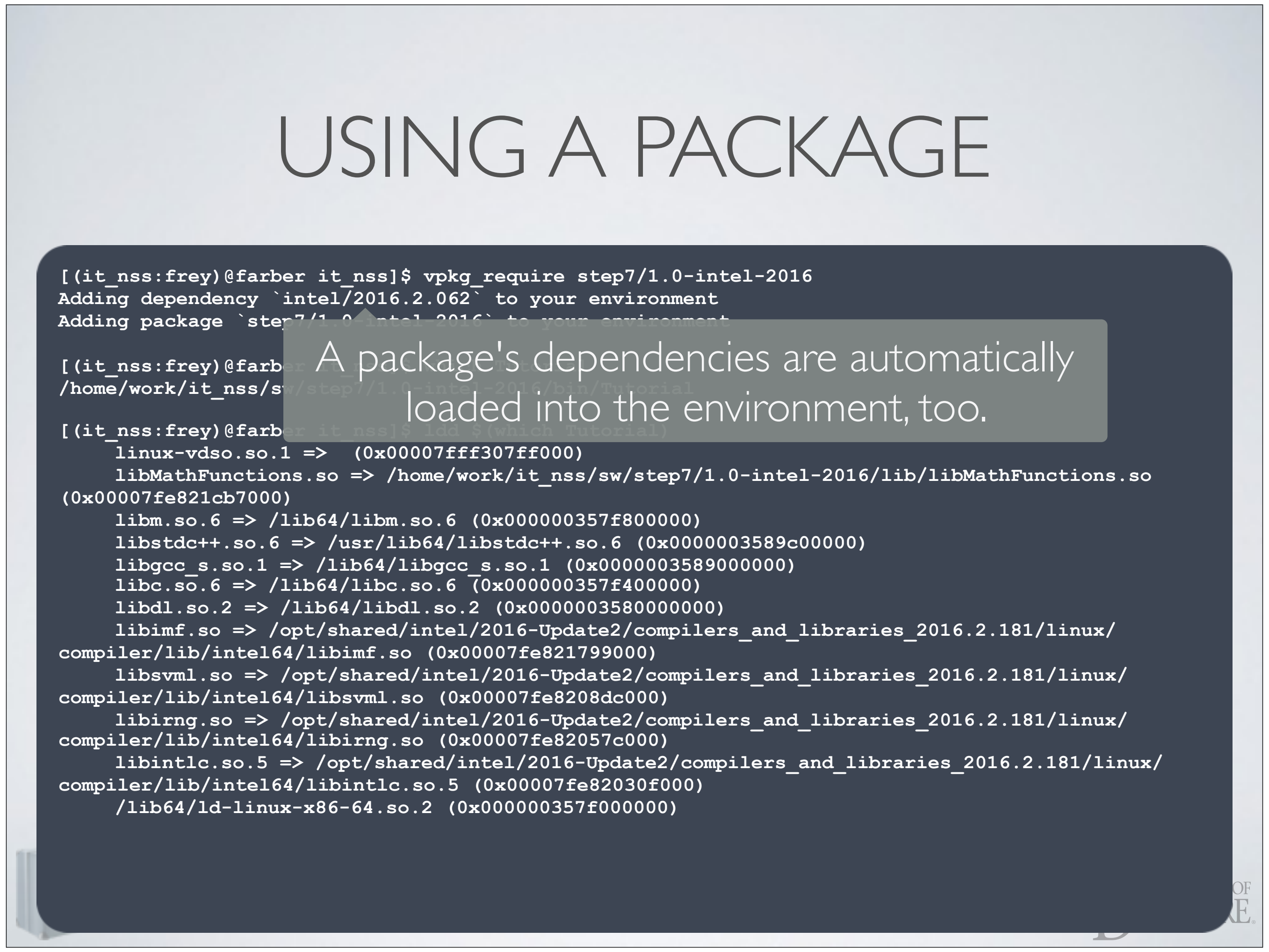
```
[(it_nss:frey)@farber ~]$ vpkg_devrequire step7/1.0

[(it_nss:frey)@farber ~]$ echo $CPPFLAGS
-I/home/work/it_nss/sw/step7/1.0/include

[(it_nss:frey)@farber ~]$ echo $LDFLAGS
-L/home/work/it_nss/sw/step7/1.0/lib
```



USING A PACKAGE

```
[(it_nss:frey)@farber it_nss]$ vpkg_require step7/1.0-intel-2016
Adding dependency `intel/2016.2.062` to your environment
Adding package `step7/1.0-intel-2016` to your environment

[(it_nss:frey)@farber it_nss]$ which Tutorial
/home/work/it_nss/sw/step7/1.0-intel-2016/bin/Tutorial

[(it_nss:frey)@farber it_nss]$ ldd $(which Tutorial)
     linux-vdso.so.1 =>  (0x00007fff307ff000)
     libMathFunctions.so => /home/work/it_nss/sw/step7/1.0-intel-2016/lib/libMathFunctions.so
(0x00007fe821cb7000)
     libm.so.6 => /lib64/libm.so.6 (0x000000357f800000)
     libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x0000003589c00000)
     libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x0000003589000000)
     libc.so.6 => /lib64/libc.so.6 (0x000000357f400000)
     libdl.so.2 => /lib64/libdl.so.2 (0x0000003580000000)
     libimf.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libimf.so (0x00007fe821799000)
     libsvml.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libsvml.so (0x00007fe8208dc000)
     libirng.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libirng.so (0x00007fe82057c000)
     libintlc.so.5 => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libintlc.so.5 (0x00007fe82030f000)
     /lib64/ld-linux-x86-64.so.2 (0x000000357f000000)
```

- The 1.0-intel-2016 version of "step7" used the Intel compiler
- Intel runtime libraries are required by the program, so LD_LIBRARY_PATH must be altered



USING A PACKAGE

```
[(it_nss:frey)@farber it_nss]$ vpkg_require step7/1.0-intel-2016
Adding dependency `intel/2016.2.062` to your environment
Adding package `step7/1.0-intel-2016` to your environment

[(it_nss:frey)@farber it_nss]$ which Tutorial
/home/work/it_nss/sw/step7/1.0-intel-2016/bin/Tutorial

[(it_nss:frey)@farber it_nss]$ ldd $(which Tutorial)
     linux-vdso.so.1 =>  (0x00007fff307ff000)
     libMathFunctions.so => /home/work/it_nss/sw/step7/1.0-intel-2016/lib/libMathFunctions.so 
(0x00007fe821cb7000)
     libm.so.6 => /lib64/libm.so.6 (0x000000357f800000)
     libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x0000003589c00000)
     libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x0000003589000000)
     libc.so.6 => /lib64/libc.so.6 (0x000000357f400000)
     libdl.so.2 => /lib64/libdl.so.2 (0x0000003580000000)
     libimf.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libimf.so (0x00007fe821799000)
     libsvml.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libsvml.so (0x00007fe8208dc000)
     libirng.so => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libirng.so (0x00007fe82057c000)
     libintlc.so.5 => /opt/shared/intel/2016-Update2/compilers_and_libraries_2016.2.181/linux/
compiler/lib/intel64/libintlc.so.5 (0x00007fe82030f000)
     /lib64/ld-linux-x86-64.so.2 (0x000000357f000000)
```

A package's dependencies are automatically loaded into the environment, too.

- The 1.0-intel-2016 version of "step7" used the Intel compiler
- Intel runtime libraries are required by the program, so LD_LIBRARY_PATH must be altered

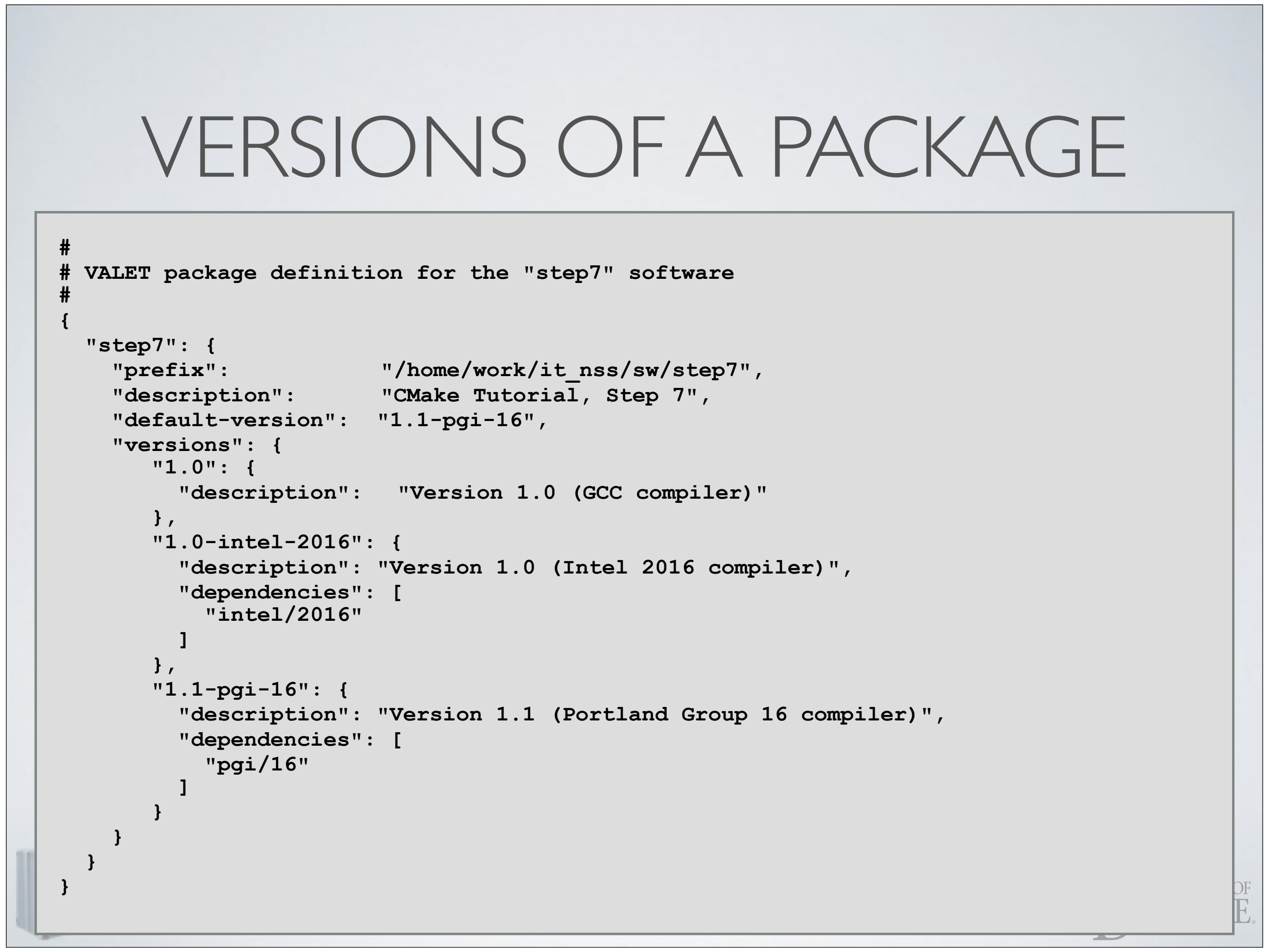
UN-USING A PACKAGE

```
[(it_nss:frey)@farber it_nss]$ vpkg_history
intel/2016.2.062
step7/1.0-intel-2016

[(it_nss:frey)@farber it_nss]$ vpkg_rollback

[(it_nss:frey)@farber it_nss]$ vpkg_history

[(it_nss:frey)@farber it_nss]$ which Tutorial
/usr/bin/which: no Tutorial in (/home/1001/bin:/opt/sbin:/home/1001/bin:/opt/sbin:/opt/bin:/opt/
shared/valet/2.0.1/bin/bash:/opt/shared/valet/2.0.1/bin:/opt/shared/univa/current/bin/lx-amd64:/
usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/ibutils/
bin)
```

UN-USING A PACKAGE

Optionally add the word "all" to remove all changes made to the current shell

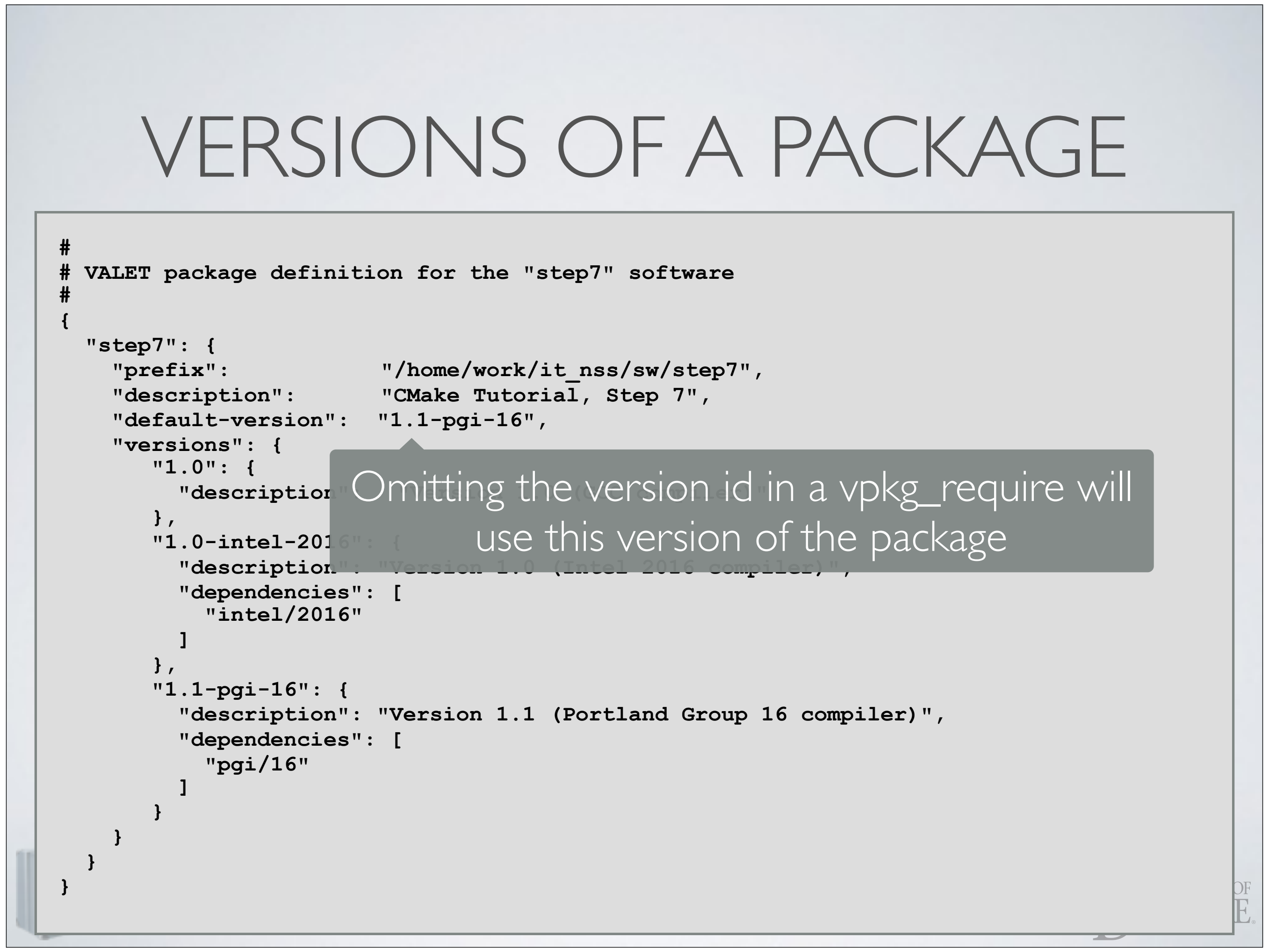
```
[(it_nss:frey)@farber it_nss]$ vpkg_history
intel/2016.2.062
step7/1.0-intel-2016

[(it_nss:frey)@farber it_nss]$ vpkg_rollback

[(it_nss:frey)@farber it_nss]$ vpkg_history

[(it_nss:frey)@farber it_nss]$ which Tutorial
/usr/bin/which: no Tutorial in (/home/1001/bin:/opt/sbin:/home/1001/bin:/opt/sbin:/opt/bin:/opt/
shared/valet/2.0.1/bin/bash:/opt/shared/valet/2.0.1/bin:/opt/shared/univa/current/bin/lx-amd64:/
usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/ibutils/
bin)
```



VERSIONS OF A PACKAGE

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":            "/home/work/it_nss/sw/step7",
    "description":       "CMake Tutorial, Step 7",
    "default-version":  "1.1-pgi-16",
    "versions": {
       "1.0": {
         "description":   "Version 1.0 (GCC compiler)"
       },
       "1.0-intel-2016": {
         "description": "Version 1.0 (Intel 2016 compiler)",
         "dependencies": [
           "intel/2016"
         ]
       },
       "1.1-pgi-16": {
         "description": "Version 1.1 (Portland Group 16 compiler)",
         "dependencies": [
           "pgi/16"
         ]
       }
    }
  }
}
```

- I've built a new copy of the "step7" program. What compiler did I use?

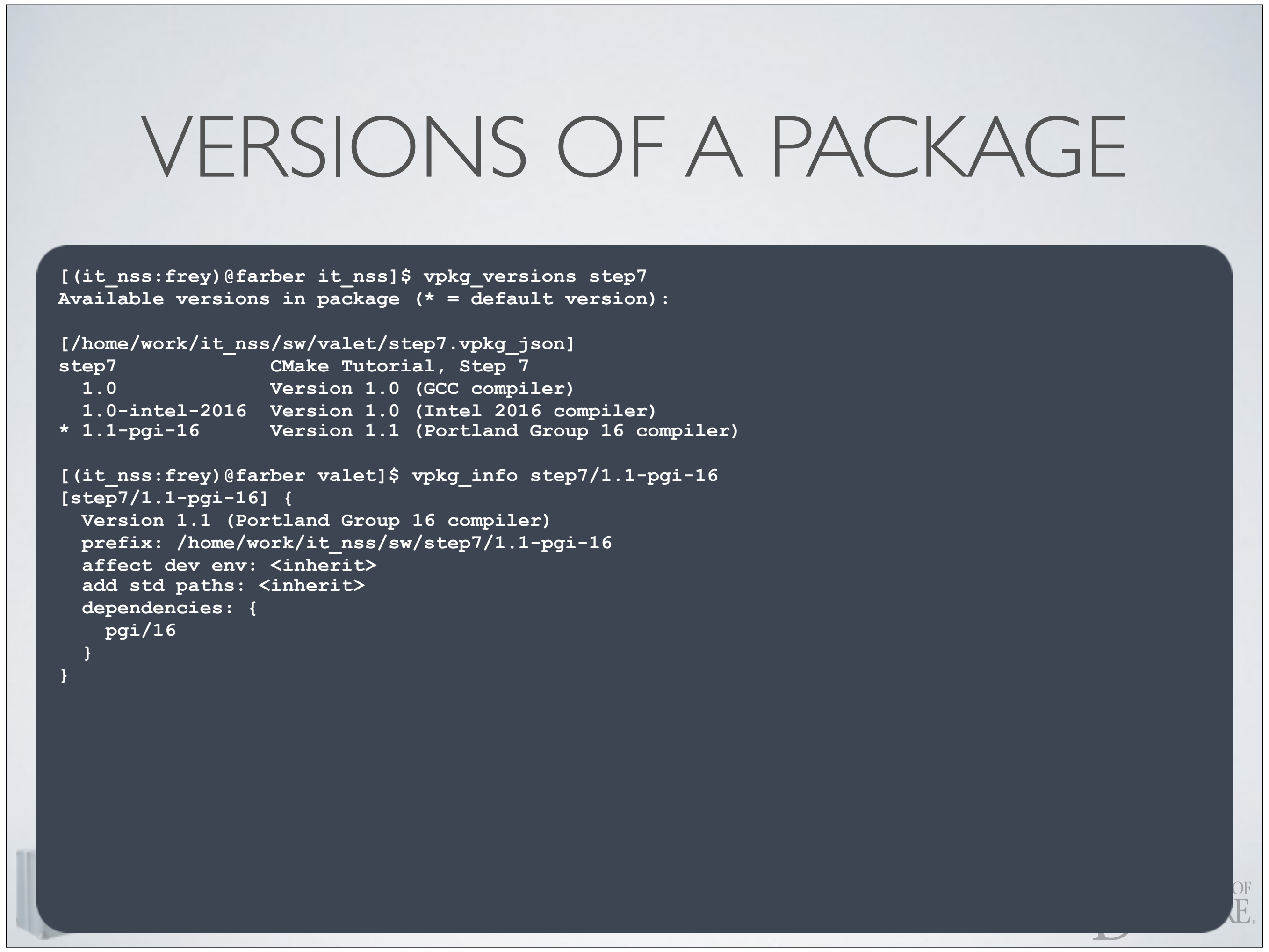


VERSIONS OF A PACKAGE

```
#
# VALET package definition for the "step7" software
#
{
  "step7": {
    "prefix":           "/home/work/it_nss/sw/step7",
    "description":      "CMake Tutorial, Step 7",
    "default-version":  "1.1-pgi-16",
    "versions": {
       "1.0": {
         "description": [illegible]
       },
       "1.0-intel-2016": {
         "description": "Version 1.0 (Intel 2016 compiler)",
         "dependencies": [
           "intel/2016"
         ]
       },
       "1.1-pgi-16": {
         "description": "Version 1.1 (Portland Group 16 compiler)",
         "dependencies": [
           "pgi/16"
         ]
       }
    }
  }
}
```

Omitting the version id in a vpkg_require will use this version of the package

- I've built a new copy of the "step7" program. What compiler did I use?



VERSIONS OF A PACKAGE

```
[(it_nss:frey)@farber it_nss]$ vpkg_versions step7
Available versions in package (* = default version):

[/home/work/it_nss/sw/valet/step7.vpkg_json]
step7             CMake Tutorial, Step 7
  1.0             Version 1.0 (GCC compiler)
  1.0-intel-2016  Version 1.0 (Intel 2016 compiler)
* 1.1-pgi-16      Version 1.1 (Portland Group 16 compiler)

[(it_nss:frey)@farber valet]$ vpkg_info step7/1.1-pgi-16
[step7/1.1-pgi-16] {
  Version 1.1 (Portland Group 16 compiler)
  prefix: /home/work/it_nss/sw/step7/1.1-pgi-16
  affect dev env: <inherit>
  add std paths: <inherit>
  dependencies: {
    pgi/16
  }
}
```

- Rather than reading the package definition file, use the vpkg_versions command
- The vpkg_info command summarizes properties of a package (or a version of a package)

ALTERNATIVE SOFTWARE LAYOUTS

- Sometimes software does not follow the compartmentalized organization
  - VASP: two separate builds in their own directories (a static library and the executable)
    - No "make install" target

Dare to be first.

ALTERNATIVE SOFTWARE LAYOUTS

- Sometimes software does not follow the compartmentalized organization
  - Create the standard directories (bin, lib, etc.) and manually copy targets into them
  - Tell VALET what paths contain executables, libraries, etc.

Dare to be first.

UNIVERSITY OF DELAWARE

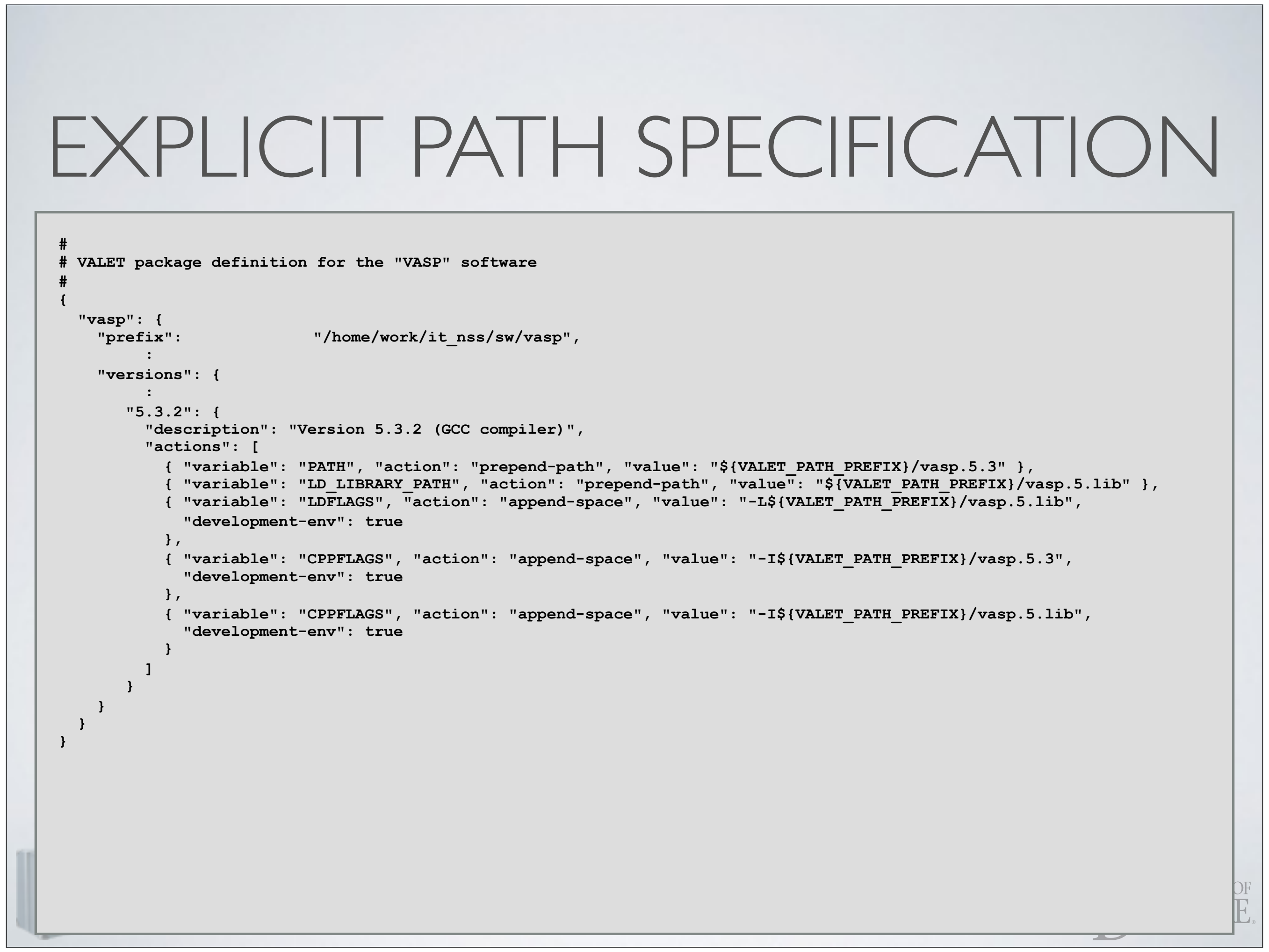
EXPLICIT PATH SPECIFICATION

```
#
# VALET package definition for the "VASP" software
#
{
  "vasp": {
    "prefix":           "/home/work/it_nss/sw/vasp",
         :
    "versions": {
         :
       "5.3.2": {
         "description": "Version 5.3.2 (GCC compiler)",
         "actions": [
           { "incdir": ["vasp.5.3", "vasp.5.lib"],
             "bindir": "vasp.5.3",
             "libdir": "vasp.5.lib"
           }
         ]
       }
    }
  }
}
```

EXPLICIT PATH SPECIFICATION

```
#
# VALET package definition for the "VASP" software
#
{
  "vasp": {
    "prefix":            "/home/work/it_nss/sw/vasp",
         :
    "versions": {
         :
       "5.3.2": {
         "description": "Version 5.3.2 (GCC compiler)",
         "actions": [
           { "incdir": ["vasp.5.3", "vasp.5.lib"],
             "bindir": "vasp.5.3",
             "libdir": "vasp.5.lib"
           }
```

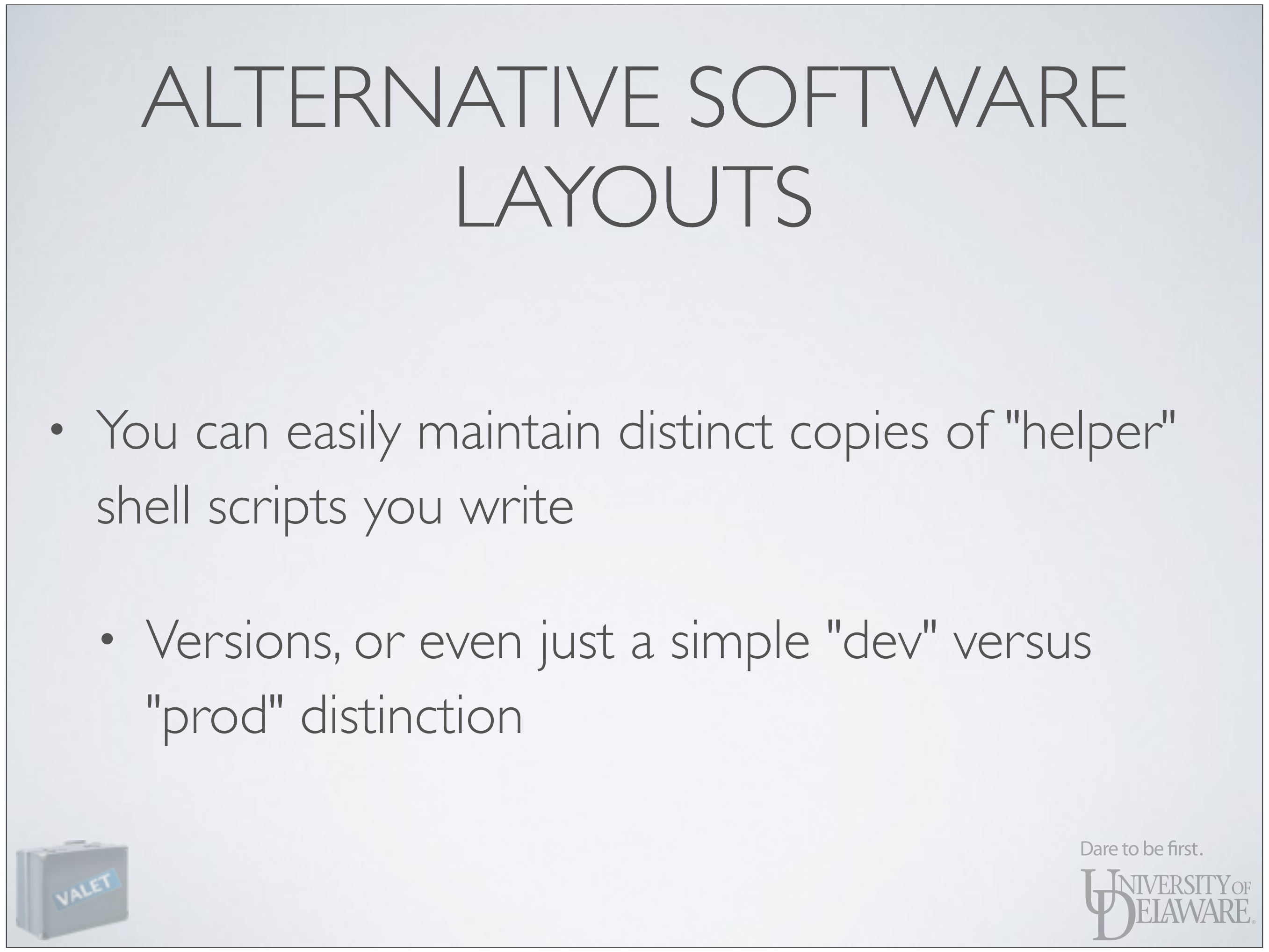
```
[(it_nss:frey)@farber ~]$ vpkg_devrequire vasp/5.3.2

[(it_nss:frey)@farber ~]$ echo $CPPFLAGS
-I/home/work/it_nss/sw/vasp/5.3.2/vasp.5.3 -I/home/work/it_nss/sw/vasp/5.3.2/vasp.5.lib

[(it_nss:frey)@farber ~]$ echo $LDFLAGS
-L/home/work/it_nss/sw/vasp/5.3.2/vasp.5.lib

[(it_nss:frey)@farber ~]$ echo $PATH
/home/work/it_nss/sw/vasp/5.3.2/vasp.5.3:/home/1001/bin:…
```



EXPLICIT PATH SPECIFICATION

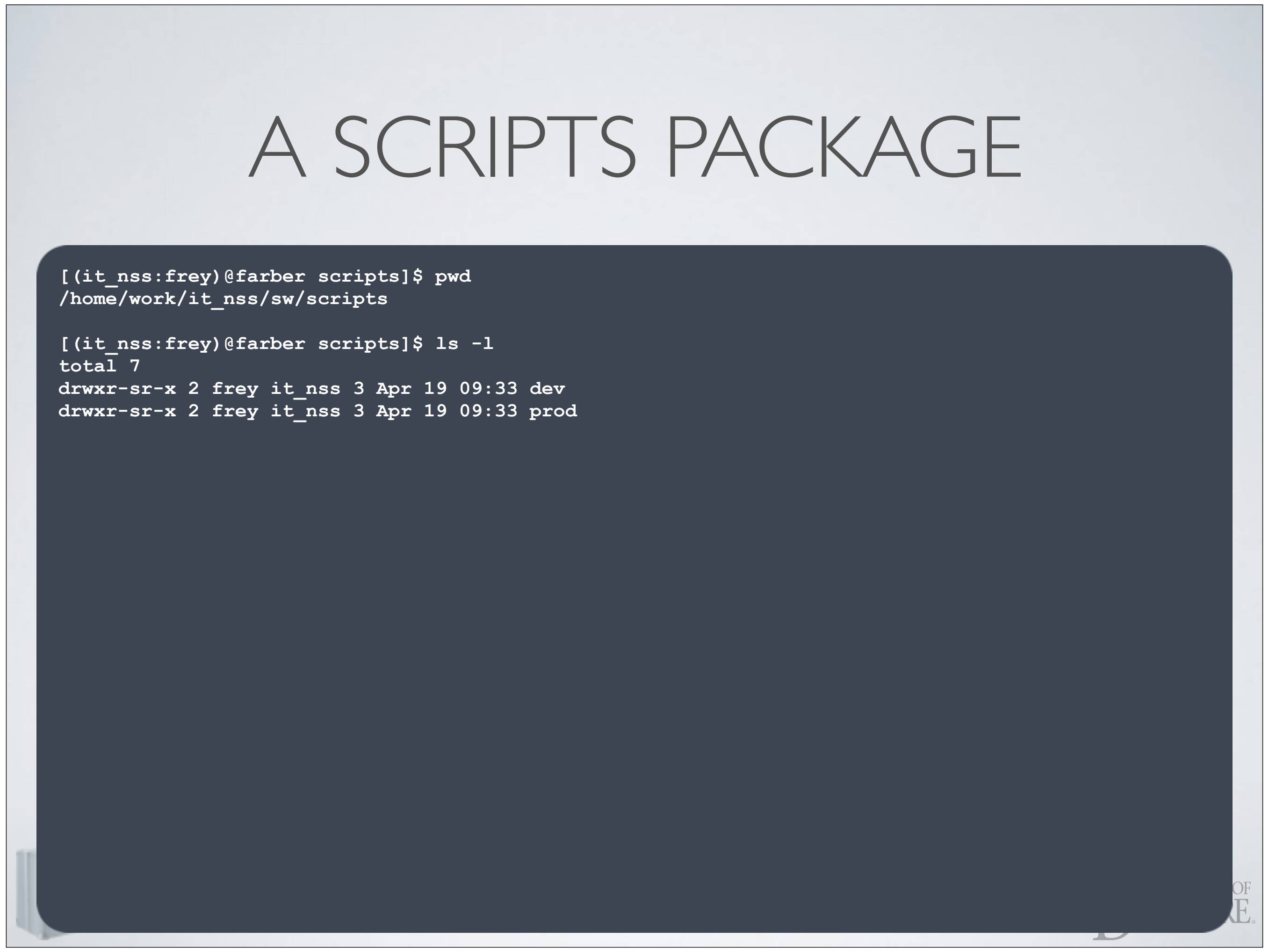
```
#
# VALET package definition for the "VASP" software
#
{
  "vasp": {
    "prefix":              "/home/work/it_nss/sw/vasp",
         :
    "versions": {
         :
       "5.3.2": {
         "description": "Version 5.3.2 (GCC compiler)",
         "actions": [
           { "variable": "PATH", "action": "prepend-path", "value": "${VALET_PATH_PREFIX}/vasp.5.3" },
           { "variable": "LD_LIBRARY_PATH", "action": "prepend-path", "value": "${VALET_PATH_PREFIX}/vasp.5.lib" },
           { "variable": "LDFLAGS", "action": "append-space", "value": "-L${VALET_PATH_PREFIX}/vasp.5.lib",
             "development-env": true
           },
           { "variable": "CPPFLAGS", "action": "append-space", "value": "-I${VALET_PATH_PREFIX}/vasp.5.3",
             "development-env": true
           },
           { "variable": "CPPFLAGS", "action": "append-space", "value": "-I${VALET_PATH_PREFIX}/vasp.5.lib",
             "development-env": true
           }
         ]
       }
    }
  }
}
```

- Equivalent to the previous method using "bindir" et al.



ALTERNATIVE SOFTWARE LAYOUTS

- You can easily maintain distinct copies of "helper" shell scripts you write
  - Versions, or even just a simple "dev" versus "prod" distinction



A SCRIPTS PACKAGE

```
[(it_nss:frey)@farber scripts]$ pwd
/home/work/it_nss/sw/scripts

[(it_nss:frey)@farber scripts]$ ls -l
total 7
drwxr-sr-x 2 frey it_nss 3 Apr 19 09:33 dev
drwxr-sr-x 2 frey it_nss 3 Apr 19 09:33 prod
```

- Prod contains scripts that have been debugged and work properly
- Dev contains scripts that are being written/modified

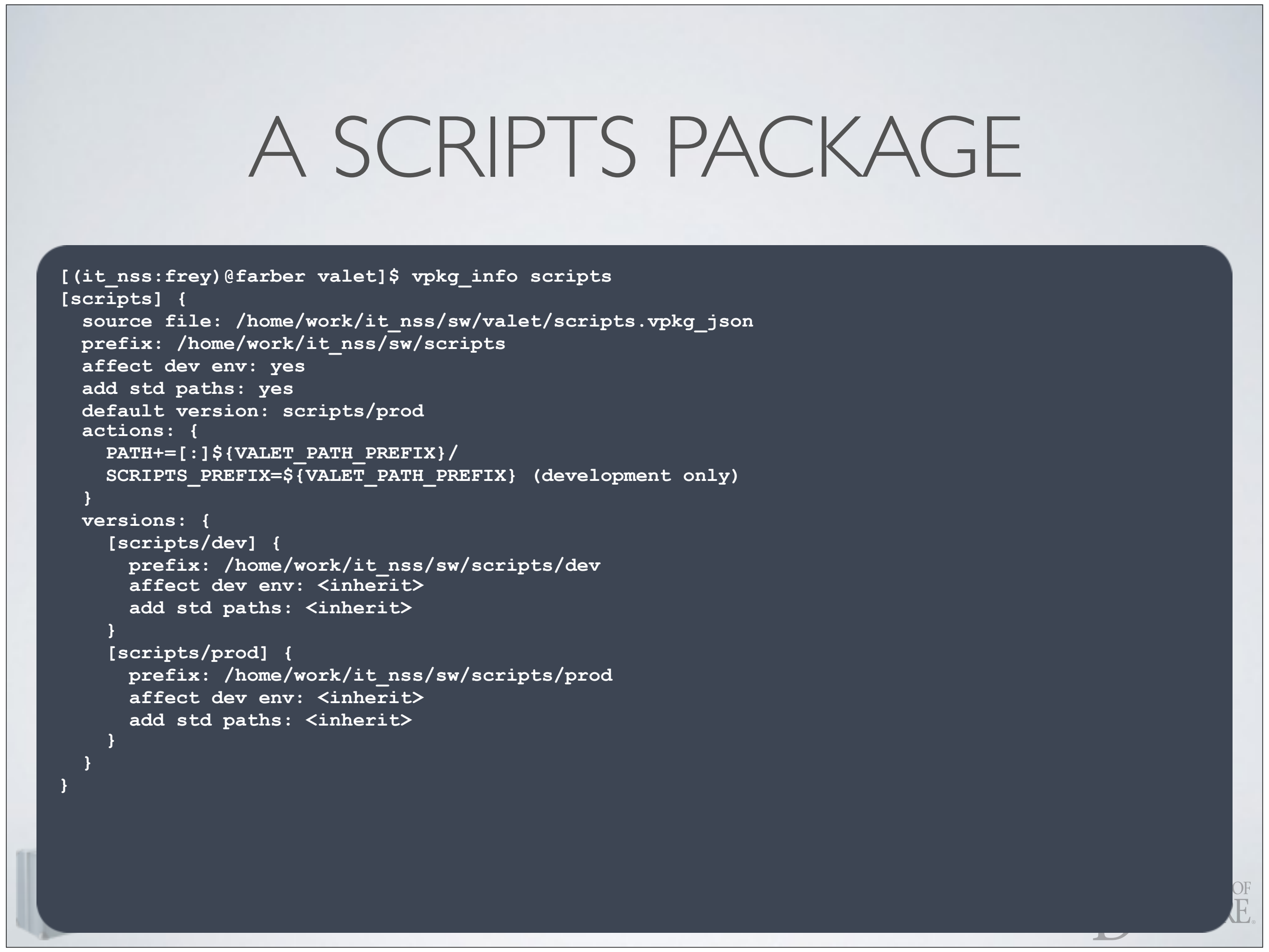
A SCRIPTS PACKAGE

```
{
  "scripts": {
    "prefix":     "/home/work/it_nss/sw/scripts",
    "default-version":  "prod",
    "actions": [
      { "bindir": "" }
    ],
    "versions": {
       "prod": {
       },
       "dev": {
       }
    }
  }
}
```

A SCRIPTS PACKAGE

```
{
  "scripts": {
    "prefix":     "/home/work/it_nss/sw/scripts",
    "default-version":  "prod",
    "actions": [
      { "bindir": "" }
    ],
    "versions": {
       "prod": {
       },
       "dev": {
       }
    }
  }
}
```

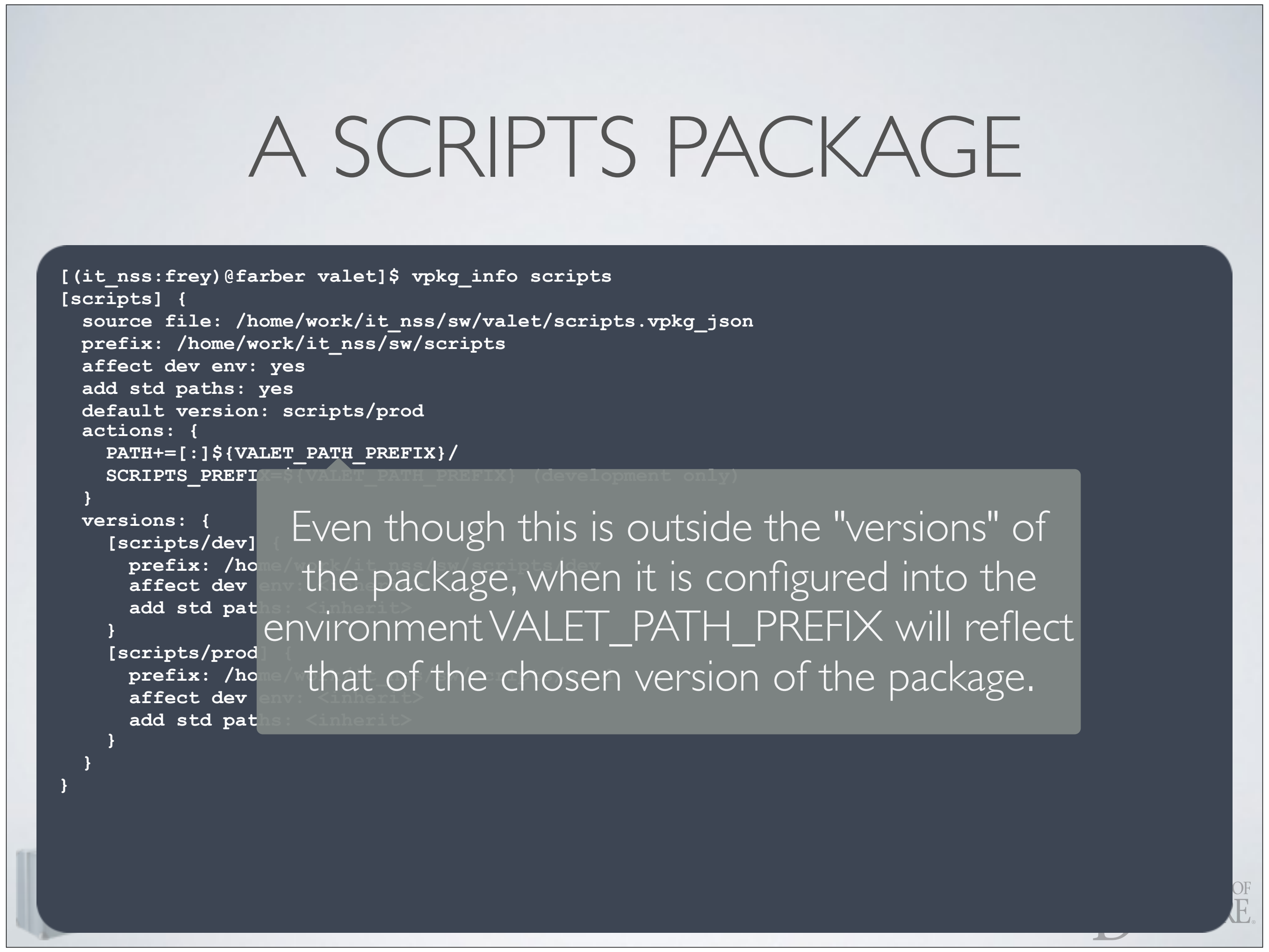
For "bindir" and friends, a relative path is relative to the VALET_PATH_PREFIX for the version of the package.



A SCRIPTS PACKAGE

```
[(it_nss:frey)@farber valet]$ vpkg_info scripts
[scripts] {
  source file: /home/work/it_nss/sw/valet/scripts.vpkg_json
  prefix: /home/work/it_nss/sw/scripts
  affect dev env: yes
  add std paths: yes
  default version: scripts/prod
  actions: {
    PATH+=[:]${VALET_PATH_PREFIX}/
    SCRIPTS_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [scripts/dev] {
      prefix: /home/work/it_nss/sw/scripts/dev
      affect dev env: <inherit>
      add std paths: <inherit>
    }
    [scripts/prod] {
      prefix: /home/work/it_nss/sw/scripts/prod
      affect dev env: <inherit>
      add std paths: <inherit>
    }
  }
}
```

- Prod contains scripts that have been debugged and work properly
- Dev contains scripts that are being written/modified



A SCRIPTS PACKAGE

```
[(it_nss:frey)@farber valet]$ vpkg_info scripts
[scripts] {
  source file: /home/work/it_nss/sw/valet/scripts.vpkg_json
  prefix: /home/work/it_nss/sw/scripts
  affect dev env: yes
  add std paths: yes
  default version: scripts/prod
  actions: {
    PATH+=[:]${VALET_PATH_PREFIX}/
    SCRIPTS_PREFIX=${VALET_PATH_PREFIX} (development only)
  }
  versions: {
    [scripts/dev] {
      prefix: /home/work/it_nss/sw/scripts/dev
      affect dev env: <inherit>
      add std paths: <inherit>
    }
    [scripts/prod] {
      prefix: /home/work/it_nss/sw/scripts/prod
      affect dev env: <inherit>
      add std paths: <inherit>
    }
  }
}
```

Even though this is outside the "versions" of the package, when it is configured into the environment VALET_PATH_PREFIX will reflect that of the chosen version of the package.

- Prod contains scripts that have been debugged and work properly
- Dev contains scripts that are being written/modified