IT 240
Shell Scripting for Administrators

Chapter 3
Lists & Arrays

Stan J. Senesy
IT Program/CCS
New Jersey Institute of Technology

Lists & Arrays

- Scalar data is singular, while lists and arrays allow multiple values to be stored in a variable
- Perl's array schema should be familiar from previous languages:

  $fred[0] = "yabba";

  $fred[1] = "dabba";

  $fred[2] = "doo";

- If your subscript exceed the bounds of the array, the extra elements are automatically created

Lists & Arrays

- A list represents comma separated values enclosed in parenthesis:

  (1, 2, 3)

  (“fred”, 4.5)

  ( )

  (1..100)

- If you have a long list of single work values and don’t wish to type a lot of “ characters, use the *qw* shortcut

Lists & Arrays

- Assignment:

  ($fred, $barney, $dino) = (“flinstone”, “rubble”, undef);

  ($fred, $barney) = ($barney, $fred);

  ($fred, $barney) = qw< flintstone rubble slate granite >;

  ($wilma, $dino) = qw[flintstone];

- You don’t need to specify every element in the array, just use the @ operator before the name

Lists & Arrays

- Try to stay away from indices when possible; your code will execute faster
- Use the pop and push operators to remove or add elements to an array
- Pop returns the last element of the array (highest subscript)
- Push adds the element to the end of the array (highest subscript)

Lists & Arrays

- To remove or add items to the beginning of an array (lowest subscript), use the shift and unshift
- Shift removes one or more elements
- Unshift adds elements

Lists & Arrays

- While it's possible to use traditional looping structures and indices to traverse an array, perl makes it much easier
- The *foreach* structure will step through a list of values taking on the value for each element in the array

Lists & Arrays

- Perl has other tools to make dealing with arrays easier
- The *reverse* operator takes a list of values and returns it in the opposite order
- The *sort* operator returns a sorted list based on ASCII char order (careful!)

`

- Context refers to where an expression is found
- When perl parses an expression, it always expects a scalar or list value
- If something is the exact same sequence of characters, it may give single, scalar or list values depending on the context!