IT 240
Shell Scripting for Administrators

Chapter 4
Subroutines

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

Subroutine Basics

- Subroutines are defined with the keyword *sub* and enclosed in braces to define the boundaries of the routine
- Although legally defined anywhere in your code, proper etiquette puts them at the start of the program
- Subroutine definitions are global and visible everywhere in your program

Subroutine Basics

- Subroutines may be invoked using the & in front of the subroutine name
- All perl subroutines have return values, whether used or not
- The evaluation of the last line of the subroutine code is returned (whether intended or not!)
- Be careful! (pgs 56/7)

Arguments

- Parameters are passed to a subroutine call by enclosing them as a list in parentheses
- Parameters are automatically stored in the special variables *$_[x]* (note that these have nothing to do with the *$[x]* series of variables
- Excess parameters are ignored

Private Variables

- By default, all perl variables are global
- The my operator may be used to define lexical (or local) variables
- These variables will have scope only in the block that they're defined in
- Arguments may be assigned to local variables in this manner:
  - *my ($m, $n) = @_;*

Enforcing Perl's Rules

- Perl tends to be a pretty permissive language; however there will be times when we wish it to play by a stricter set of rules
- The *use strict pragma* sets the compiler to enforce rules of good programming etiquette
- Among other things, this forces you to declare variables, etc.

Many Happy Returns

- Suppose you want something other than the last line of a subroutine returned?
- Perl will immediately return a value when the *return* operator is called
- As soon as it's called, the focus exits from the subroutine!

Ampersand Optional?

- If perl can recognize that a call is to a subroutine (by it's parameter list), the & character is optional
- If your subroutine has the same name as a perl built-in function (a bad idea), then you MUST use the & to identify your subroutine no matter what

Final Thoughts

- Unlike many other languages, perl is not limited to just returning a scalar variable
- Perl is perfectly happy returning a list or other non-scalar value without the headaches required of languages like C or C++