

# CS306: Introduction to Perl

## Hashes

U. of Alabama at Birmingham  
Dept. of Computer & Information Sciences

Slide 1

# Introduction to Hashes

- A *hash* is similar to an array, but it is indexed by strings instead of numbers.
- Each element of a hash has a *key* and a *value*
- Each key is a string and is unique
- Each value is a scalar

Slide 2

## Creating Hashes

- Perl uses the `%` symbol to indicate a hash
- `%carcolor = ("Fran", "blue", "Stacy", "white");`
  - Hash assignments expect a list consisting of name-value pairs (so the list should have an even number of elements)
- `%profile` is not related to `@profile`, which is not related to `$profile`

Slide 3

## Using The `=>` Big Arrow Notation

- Perl provides the `=>` notation to make hash creation easier. It's called the "big arrow"
- `%carcolor = ("Fran" => "blue", "Stacy" => "white");`
  - Easier to see the name => value pairs this way

Slide 4

## Hashes <--> Arrays

- Since both hashes and arrays are built from lists, you can interchange them
- `@array = qw/Fran blue Stacy white/;`
- `%carcolor = @array;`
  - Only works when `@array` has even number of elements
- `@array = %carcolor; # key/val in random order`

Slide 5

## Accessing Hash Elements

- Use the `{key}` notation to lookup a value
- `$carcolor{Fran} # returns "blue"`
  - `$carcolor{"Fran"}` is fine too
- `$hash{$key} # where $key holds a key name`
- `$name = "Fran";`  
`print "$name has a $carcolor{$name} car.\n";`

Slide 6

## Modifying Hashes

- Add a single element
  - `$carcolor{Jane} = "black";`
- Change a single element
  - `$carcolor{Stacy} = "green";`
- Delete an element
  - `delete $carcolor{Stacy};`

Slide 7

## Neatly viewing a hash

- At the top of your program:  
`use Data::Dumper;`
- Then, where you want to print the hash:  
`print Dumper(%myhash);`
- This will print a neatly formatted representation of your hash

Slide 8

## Hash Functions

- *keys()* - returns a list of the keys in the hash
  - @keys = keys %hash;
- *values()* - returns a list of the values in the hash
  - @values = values %hash;
  - Generally want to avoid this – once you call values(), you've broken the association between the key and value, which was the whole reason we had a hash

Slide 9

## Hash Functions 2

- *each()* - pop for hashes. Pops off one key => value pair per call.
  - (\$key, \$value) = each %hash;
  - This makes more sense in loops...

```
while (($key, $value) = each %hash) {  
    # Do something with the key and value  
}
```
  - How does the while loop know when to stop? It's subtle...

Slide 10

## Hash Functions 3

- *keys()*, *values()* and *each()* do not return their results in any well-defined order.
- Use *sort()* for that
  - @keys = sort keys %hash;
  - As always, *sort(keys(%hash))* is equivalent. Good to remind oneself every now and then.

Slide 11

## Hash Functions 4

- *exists()* - returns true if particular key exists
  - if (exists \$hash{"somekey"}) ...
- *delete()* - removes a key => value pair
  - delete \$hash{\$otherkey};
  - Why doesn't this work...
    - \$hash{\$otherkey} = undef;

Slide 12

## Hash Functions 5

- `reverse()` - flips all of the keys with their values
- ```
%iplookup = ('blazer1' => '138.26.65.80',  
             'blazer2' => '138.26.65.81',  
             'blazer3' => '138.26.65.82');
```
- “I have this `$ip`, but what's the host name?”
  - ```
%hostlookup = reverse %iplookup;  
print $hostlookup{$ip};
```
- How is this broken?

Slide 13

## Iterating Over Hashes

- ```
foreach $key (sort keys %hash) {  
    print "The value of $key is $hash{$key}.\n";  
}
```
- ```
while (($key, $value) = each %hash) {  
    # Do something with the key and value  
}
```

Slide 14

## Uses For Hashes

- Hashes are one of Perl's best features, they are extremely versatile and powerful.
- Typical uses...
  - Lookup tables like the host/ip example
  - Tracking the same data for many people/things
    - ```
$diskusage{'fran'} = 200; $diskusage{'ying'} = 50;
```
  - Counters
    - ```
for $score (@scores) { $results{$score}++; }  
# %results holds how many people got each specific score
```

Slide 15