Decisions
if Command

• if statement enables you to test a condition and then change the flow of program execution based on the result of the test.

• The general format of the if command is

if \( command_t \)
then
  \( command \)
  \( command \)
  ...
fi

where \( command_t \) is executed and its exit status is tested. If the exit status is zero, the commands that follow between the then and the fi are executed; otherwise, they are skipped.
Exit Status

• Whenever any program completes execution under the Unix system, it returns an exit status back to the system.
• This status is a number that usually indicates whether the program successfully ran.
• By convention, an exit status of zero indicates that a program succeeded, nonzero indicates that it failed.
• For example, the `cp`, `grep` 
• In a pipeline, the exit status is that of the last command in the pipe. For example, the `who | grep fred`
The `$?` Variable

- The shell variable `$?` is automatically set by the shell to the exit status of the last command executed.
- Naturally, you can use `echo` to display its value at the terminal.

```bash
$ echo $? ```
A shell program called on that tells us whether a specified user is logged on to the system. The username of the user to check will be passed to the program on the command line as an argument. If the user is logged on, a message will be printed to that effect. Otherwise the program do nothing.

- System’s “garbage can”
- Regular expression for grep
The test Command

• A built-in shell command called test is most often used for testing one or more conditions in an if command.

• Its general format is test expression where expression represents the condition you are testing.

• test evaluates expression, and if the result is true, it returns an exit status of zero; otherwise, the result is false, and it returns a nonzero exit status.
String Operators in Expression

- **test String Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Returns TRUE (exit status of 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string1 = string2</code></td>
<td><code>string1</code> is identical to <code>string2</code>.</td>
</tr>
<tr>
<td><code>string1 != string2</code></td>
<td><code>string1</code> is <em>not</em> identical to <code>string2</code>.</td>
</tr>
<tr>
<td><code>string</code></td>
<td><code>string</code> is not null.</td>
</tr>
<tr>
<td><code>-n string</code></td>
<td><code>string</code> is not null (and <code>string</code> must be seen by test).</td>
</tr>
<tr>
<td><code>-z string</code></td>
<td><code>string</code> is null (and <code>string</code> must be seen by test).</td>
</tr>
</tbody>
</table>
An Alternate Format for test

• This format improves the readability of the command, especially when used in if commands.
  test expression
  [ expression ]
Integer Operators in Expression

- **test Integer Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Returns TRUE (exit status of 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int1 -eq int2</code></td>
<td><code>int1</code> is equal to <code>int2</code>.</td>
</tr>
<tr>
<td><code>int1 -ge int2</code></td>
<td><code>int1</code> is greater than or equal to <code>int2</code>.</td>
</tr>
<tr>
<td><code>int1 -gt int2</code></td>
<td><code>int1</code> is greater than <code>int2</code>.</td>
</tr>
<tr>
<td><code>int1 -le int2</code></td>
<td><code>int1</code> is less than or equal to <code>int2</code>.</td>
</tr>
<tr>
<td><code>int1 -lt int2</code></td>
<td><code>int1</code> is less than <code>int2</code>.</td>
</tr>
<tr>
<td><code>int1 –ne int2</code></td>
<td><code>int1</code> is not equal to <code>int2</code></td>
</tr>
</tbody>
</table>
File Operators in Expression

- **Commonly Used** test File Operators
  
<table>
<thead>
<tr>
<th>Operator</th>
<th>Returns TRUE (exit status of 0) if</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-d file</code></td>
<td><code>file</code> is a directory.</td>
</tr>
<tr>
<td><code>-e file</code></td>
<td><code>file</code> exists.</td>
</tr>
<tr>
<td><code>-f file</code></td>
<td><code>file</code> is an ordinary file.</td>
</tr>
<tr>
<td><code>-r file</code></td>
<td><code>file</code> is readable by the process.</td>
</tr>
<tr>
<td><code>-s file</code></td>
<td><code>file</code> has nonzero length.</td>
</tr>
<tr>
<td><code>-w file</code></td>
<td><code>file</code> is writable by the process</td>
</tr>
</tbody>
</table>
The Logical Operators in Expression

- The logical negation operator `!`
- The logical AND operator `-a`
- The logical OR operator `-o`
- Parentheses
The else Construct in if

• A construct know as the else can be added to the if command, with the general format as show:
  if $command_t$
  then
    $command$
    $command$
    $command$
    ...
  else
    $command$
    $command$
    $command$
    ...
  fi

• Modified version of on adding else
• Modified version of on checking the number of arguments
The exit Command

• A build-in shell command called exit enables you to immediately terminate execution of your shell program.

• The general format of this command is
  `exit n`
  where `n` is the exit status that you want returned. If none is specified, the exit status used is that of the last command executed before the exit.

• Execute the exit command directly from your terminal will log you off the system because it will have the effect of terminating execution of your login shell.
A second Look at the rem Program

• Check the number of arguments
• Use exit to exit early from the program if needed.
The elif Construct

• If \( \text{command}_t \) then
  command
  command
  ... 
else
  if \( \text{command}_t \) then
    command
    command
    ... 
  else
    ... 
    if \( \text{command}_t \) then
      command
      command
    else
      command
      command
    fi
    fi
  fi
fi
• greetings program
  suppose that you wanted to write a program called greetings that would print a friendly “Good morning,” “Good afternoon,” or “Good evening” when you run it, considering any time from midnight to noon to be the morning, noon to 6:00 p.m. the afternoon, and 6:00 p.m. to midnight the evening.