Chapter 12

Debugging Techniques

Overview

- Interacting with user
- Using Standard Error
- Options for Debugging
- Conditional Debugging

Lesson: Interacting with user

- Write to screen

```
echo "DEBUG: made to this far"
echo "DEBUG: filename is $filename"
```

- To pause script execution
  - `sleep` – pause temporarily

```
echo "DEBUG: ok so far, Filename is [$Filename]"
sleep 3    # Sleep for 3 seconds ...
```

  - `read` – wait for input

```
echo "DEBUG: Filename is [$Filename], press RETURN"
read    # Read line to the default variable REPLY
```

Lesson: Using Standard Error

- If You process script's output, sending debug messages to `stdout` will mix script output and debug messages.
  - Use `stderr` instead:

```
if [ ! -r "$File" ]; then
  echo "Warning: File $File not readable." 1>&2
fi

{
  echo "This goes to stderr."
  echo "So does this line!"
} 1>&2
```

Lesson: Options for Debugging

- Shell options
- Script Tracing

Shell options

- There are several shell options useful for debugging
- `-n` – noexec. Check syntax only

```
$ sh -n setx
$ sh -n setx.csh
setx.csh[5]: syntax error: 'if' unmatched

$ cat myscript
set -n
echo "there we are"
$ ./myscript
```

- `-v` – verbose. Send script lines to `stderr`, when read

```
$ sh -v myscript
set -o vi
set -n
echo "there we are"
```

Script Tracing

- Trace mode shows every command before running
- Commands are showed as executed
  - Helps when there are many aliases and substitutions

```
$ cat setx
...
set -x
echo $0
...

$ ./setx
+ echo ./setx
./setx
...
```

Lesson: Conditional Debugging

- Conditional Debugging
- Shell signals

Conditional Debugging

- For larger scripts and projects it might be good idea to implement explicit debug mode.

```
while getopts d VAR
do
  case $VAR in
    d) debugmode=Y ;;
  esac
done
shift $((OPTIND-1))
...
if [ "Y" -eq "$debugmode"]; then
  {
    echo "DEBUG: $0 starts"
    echo "DEBUG: filename is $file"
  } 1>&2
  set -o xtrace
fi
```

Shell signals

- **Korn family shells have signals:**
  - DEBUG signal

```
trap "echo $0: [\$LINENO]" DEBUG
```

  - ERR signal

```
trap "echo \"$0: Error, line \$LINENO;
  cmd returned \$?\"" ERR
```

  - RETURN signal

```
trap 'echo "returning from the function"' RETURN
```

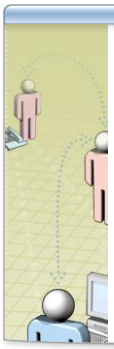
  - EXIT signal

```
trap 'echo "exiting from the script"' EXIT
```

Review Exercises

- **Complete the exercises from the Learning Guide**

Topics for Review



1. **Read the review topics**
2. **Think about what you learned in this Session in the context of your own work environment**
3. **Discuss your answers as a class**