

# *Perl Training*



## Cool things to do in Perl

<http://perlwizard.org/perl/week13>

# *Agenda*

- Homework Review
- Signals
- Homework!!!

# *Signal Processing*

Signals are a way to communicate things to a script already running

Examples are the ability to turn debugging on/off without killing and restarting the script. Another is to change the functionality of an already running script.

## *%SIG hash*

If you want your perl script to listen to signals, you simply need to define references in the %SIG hash

```
# On Interrupt signal, print "Exiting" and exit
```

```
$SIG{INT} = sub { die "\nExiting\n"; }
```

```
# When the alarm signal is received, turn on the buzzer
```

```
$SIG{ALRM} = sub { die "Bzzzzzzzzzz\n"; }
```

```
# When the USR1 signal is received, turn on debugging messages
```

```
$SIG{USR1} = sub { $debug = 1; }
```

# *Annoying Perl Programs*

```
#!/usr/bin/perl
# Annoying little program that is (somewhat) hard to kill
sub catch_signal {
    print "I am invincible!!!\n";
}
$SIG{INT} = \&catch_signal;
$SIG{QUIT} = \&catch_signal;
while(1) { print ":p\n"; sleep(1);}
```

# *Ignoring Interrupts*

If part of your program absolutely has to finish, and should not be interrupted, use the following:

```
...  
{  
    local $SIG{INT} = 'IGNORE';  
    ...      # Do what you need to here, ignoring interrupts  
}          # End of block turns interrupts back on
```

# *Sending Signals to yourself*

```
kill(HUP, $$);
```

```
kill(USR1, $$);
```

```
# Want to see all the signals available?
```

```
@allKeys = keys(%SIG);
```

```
print (join(" ", @allKeys));
```

```
# Prints XCPU, ILL, QUIT, STOP, EMT, ABRT, BUS, USR1, XFSZ, TSTP, INT, IOT, USR2,  
# INFO, TTOU, ALRM, KILL, HUP, URG, PIPE, CONT, SEGV, VTALRM, PROF, TRAP,  
# IO, TERM, WINCH, CHLD, FPE, TTIN, SYS
```



# *Homework*

- Write a quiz program of some sort (give a time limit for each question)
- Write a program that toggles debugging when it receives a USR1 signal

*Questions?*

