Perl Examples

---

1. Calling System Commands
    - Image Processing
    - Renaming Files
    - File Conversion
    - Creating Directories
    - Padding & Unpadding Files

2. Scanning the Network
    - Finding Free Machines
    - Finding Processes
    - Finding Files
    - Finding Users

3. File Manipulation
    - Generating HTML Files
    - Generating Xpost Scripts
    - Modifying Files
    - Convert Raw Data

Remember: In order to be able to run your perl script, it must begin with the line:

```
#!/usr/local/bin/perl
```

Furthermore, if you've named the file "myFile", then to make the file executable, you need to type in a unix window:

```
chmod 755 myFile
```

Image Processing

```
#!/usr/local/bin/perl
#
# composite series of images over a background image
#

if ($#ARGV != 4) {
 print "usage: compem bg.rgb inbase outbase startNum stopNum\n";
 exit;
}
```

```
$bg = $ARGV[0];
$inbase = $ARGV[1];
$outbase = $ARGV[2];
$start = $ARGV[3];
$stop = $ARGV[4];

# for each image
for ($i=$start; $i <= $stop; $i++) {

    # pad numbers
    $num = $i;
    if($i<10) { $num = "00$i"; }
    elsif($i<100) { $num = "0$i"; }

    # call unix command "over"
    $cmd = "over $bg $inbase.$num $outbase.$num 0 0";
    print $cmd."\n";
    if(system($cmd)) { print "over failed\n"; }
}
```

Renaming Files

```
#!/usr/local/bin/perl
#
# rename series of frames
#
if ($#ARGV != 3) {
    print "usage: rename old new start stop\n";
    exit;
}

$old = $ARGV[0];
$new = $ARGV[1];
$start = $ARGV[2];
$stop = $ARGV[3];

for ($i=$start; $i <= $stop; $i++) {

    $num = $i;
    if($i<10) { $num = "00$i"; }
    elsif($i<100) { $num = "0$i"; }

    $cmd = "mv $old.$num $new.$num";
    print $cmd."\n";
    if(system($cmd)) { print "rename failed\n"; }
}
```

File Conversion

```
#!/usr/local/bin/perl
#
# convert series of images from one format to another
#
if ($#ARGV != 5) {
    print "usage: fconvert intype outtype old new start stop\n";
    exit;
}

$intype = $ARGV[0];
$outtype = $ARGV[1];
$old = $ARGV[2];
$new = $ARGV[3];
$start = $ARGV[4];
$stop = $ARGV[5];

for ($i=$start; $i <= $stop; $i++) {

    $num = $i;
    if($i<10) {$num = "00$i"; }
    elsif($i<100) { $num = "0$i"; }

    $cmd = "imgcvt -i $intype -o $outtype $old.$num $new.$num";
    print $cmd."\n";
    if(system($cmd)) { print "imgcvt failed\n"; }
}
```

Creating Directories

```
#!/usr/local/bin/perl
#
# create a series of directories
#
if ($#ARGV != 2) {
    print "usage: mkdirs base start stop\n";
    exit;
}

$base = $ARGV[0];
$start = $ARGV[1];
$stop = $ARGV[2];

for ($i=$start; $i <= $stop; $i++) {

    $num = $i;
    if($i<10) {$num = "00$i"; }
    elsif($i<100) { $num = "0$i"; }

    $cmd = "mkdir $base$num";
    print $cmd."\n";
    if(system($cmd)) { print "mkdir failed\n"; }
}
```

Padding & Unpadding Files

```
#!/usr/local/bin/perl
#
# pad file numbers with zeros
#
if ($#ARGV != 2) {
    print "usage: pad base start stop\n";
    exit;
}

$base = $ARGV[0];
$start = $ARGV[1];
$stop = $ARGV[2];

for ($i=$start; $i <= $stop; $i++) {

    $num = $i;
    if($i<10) {$num = "00$i"; }
    elsif($i<100) { $num = "0$i"; }

    $cmd = "mv $base$i $base$num";

    # to unpad, use this instead:
    # $cmd = "mv $base$num $base$i";

    print $cmd."\n";
    if(system($cmd)) { print "pad failed\n"; }
}
```

Finding Free Machines

```
#!/usr/local/bin/perl
#
# search list of machines for machines with no users logged on
#
$machines = `systems sgi`;
chop($machines);
@sgis = split(/ /, $machines);
@sgis = sort(@sgis);

foreach $machine (@sgis) {

    if(!(`rusers $machine`)) {
        print "$machine\n";
    }
}
```

Finding Processes

```
#!/usr/local/bin/perl
#
# search for processes running on machines
#

if ($#ARGV != 0) {
    print "usage: findprocess process\n";
    exit;
}

$process = $ARGV[0];
$machines = `systems sgi`;
chop($machines);
@sgis = split(/ /,$machines);
@sgis = sort(@sgis);

foreach $machine (@sgis) {

    print "Checking $machine...\n";

    @lines = `rsh $machine \"ps -ef | grep $process | grep -v findprocess |
grep -v grep\"`;

    if(@lines) {
        foreach $line (@lines) {
            $line =~ /^\s*(\w+)\s+(\d+)/;
            $user = $1;
            $pid = $2;
            print "$user on $machine  pid: $pid\n";
        }
    }
}
```

Finding Files

```
#!/usr/local/bin/perl
#
# search for a file in all subdirectories
#
if ($#ARGV != 0) {
    print "usage: findfile filename\n";
    exit;
}

$filename = $ARGV[0];

# look in current directory
$dir = `pwd`;
chop($dir);
&searchDirectory($dir);
```

```
sub searchDirectory {
    local($dir);
    local(@lines);
    local($line);
    local($file);
    local($subdir);

    $dir = $_[0];

    # check for permission
    if(-x $dir) {

        # search this directory
        @lines = `cd $dir; ls -l | grep $filename`;
        foreach $line (@lines) {
            $line =~ /\s+(\S+)$/;
            $file = $1;
            print "Found $file in $dir\n";
        }

        # search any sub directories
        @lines = `cd $dir; ls -l`;
        foreach $line (@lines) {
            if($line =~ /^d/) {
                $line =~ /\s+(\S+)$/;
                $subdir = $dir."/".$1;
                &searchDirectory($subdir);
            }
        }
    }
}
```

Finding Users

```
#!/usr/local/bin/perl
#
# check whether user is logged on
#

if ($#ARGV != 0) {
    print "usage: finduser username\n";
    exit;
}

$username = $ARGV[0];
$machines = "insanity ".`systems sgi`;
chop($machines);
@machines = split(/ /,$machines);
@machines = sort(@machines);

foreach $machine (@machines) {
```

```
    if(`rusers $machine | grep $username`) {
        print "$username logged on $machine\n";
    }
}
```

Generating HTML Files

```
#!/usr/local/bin/perl
#
# create n html files linked together in slide show
#

if ($#ARGV != 1) {
    print "usage: htmlslides base num\n";
    exit;
}

$base = $ARGV[0];
$num = $ARGV[1];

for ($i=1; $i <= $num; $i++) {

    open(HTML, ">$base$i.html");

    if($i==$num) {
        $next = 1;
    } else {
        $next = $i+1;
    }

    print HTML "<html>\n<head>\n<title>$base$i</title>\n</head>\n<body>\n";
    print HTML "<a href=\"$base$next.html\"><img src=\"$base$i.jpg\"></a>\n";
    print HTML "</body>\n</html>\n";

    close(HTML);
}
```

Generating Xpost Scripts

```
#!/usr/local/bin/perl
#
# generate an xpost script to adjust saturation, and run xpost
#
if ($#ARGV != 2) {
 print "usage: fixsat infile.tiff outfile.tiff satval\n";
 exit;
}

$infile = $ARGV[0];
$outfile = $ARGV[1];
```

```
$satval = $ARGV[2];

# open xpost script
open(XPOST, ">__tmp.xp");

# set view to register A
print XPOST "view A\n";

# load original image into reg A
print XPOST "load $infile\n";

# run Kmult to turn down saturation
print XPOST "Kmult $satval $satval $satval  1.0 a b\n";

# set view to register B
print XPOST "view B\n";

# save unsaturated image
print XPOST "save tiff $outfile\n";

# close xpost script
close(XPOST);

# run xpost script
$cmd = "xpost -q -s __tmp.xp";
print $cmd."\n";
system($cmd);

# clean up
$cmd = "/bin/rm -f __tmp.xp";
print $cmd."\n";
system($cmd);
```

Modifying Text Files

```
#!/usr/local/bin/perl
#
# change all occurances of a string in a file to another string
#

if ($#ARGV != 3) {
    print "usage: chstring oldfile newfile oldstring newstring\n";
    exit;
}

$oldfile = $ARGV[0];
$newfile = $ARGV[1];
$old = $ARGV[2];
$new = $ARGV[3];

open(OF, $oldfile);
open(NF, ">$newfile");
```

```
# read in each line of the file
while ($line = <OF>) {
    $line =~ s/$old/$new/;
    print NF $line;
}

close(OF);
close(NF);
```

Convert Raw Timecode Data to Readable Data

```
#!/usr/local/bin/perl
#
# Change raw timecode data to different format
#
# timecode data event looks like:
#
# Event: 1
# 00:01:05:23
# 00:01:27:21
# a-2-9
#
# Event: 2
# 00:01:56:13
# 00:02:03:19
# a-3-9
#
# ...and so on...
#
# Want to change it to the form:
#
# a-2-9 = 21.93 seconds = 658 frames
# a-3-9 = 7.20 seconds = 216 frames
#

open(FP,"<log.txt");

$first = 1;
$total = 0;

while($line = <FP>) {

   if ($line =~ /^\d\d/ && $first) {
       $in = $line;
       $first = 0;
    } elsif ($line =~ /^\d\d/ && !$first) {
       $out = $line;
       $first = 1;
    } elsif ($line =~ /^\w-/) {
       $shot = $line;
       chop($shot);

       # parse timecodes and
```

```
        # translate in and out into seconds
        $in =~ /(\d\d):(\d\d):(\d\d):(\d\d)/;
        $hrs = $1;
        $mns = $2;
        $scs = $3;
        $fms = $4;
        $inSecs = $hrs * 3600 + $mns * 60 + $scs + $fms / 30;

        $out =~ /(\d\d):(\d\d):(\d\d):(\d\d)/;
        $hrs = $1;
        $mns = $2;
        $scs = $3;
        $fms = $4;
        $outSecs = $hrs * 3600 + $mns * 60 + $scs + $fms / 30;

        # calc duration
        $dur = $outSecs - $inSecs;
        $total += $dur;

        # print line
        printf("$shot = %.2f seconds = %d frames\n", $dur, $dur * 30);
    }
}

print "total = ".($total / 60)." mins\n";

close FP;
```