Sample: Perl - File Parser

```
#!/usr/bin/perl -w

# define variables with names of fiels
my $KEYWORDS_FILE = "keywords.txt";
my $LOG_FILE = "syslog";
my $OUT_FILE = "output-inter.txt";

# define array for keywords
my @words;

# open file with keywords or exit and print error message
open(KWF, "< $KEYWORDS_FILE") or die "Can't open $KEYWORDS_FILE for read: $!";
# for all lines in keywords file
while (my $word = <KWF>){
        # delete "\n" character from the end of the line
   chomp $word;
   # add word to words array
   # (assume that keywords.txt contains one word per line)
   push (@words, $word);
}
# close file with keywords
close KWF or die "Cannot close $KEYWORDS_FILE: $!";

# open syslog file for reading
open(LF, "< $LOG_FILE") or die "Can't open $LOG_FILE for read: $!";
# create or truncate output file
open(OF, "> $OUT_FILE") or die "Can't open $OUT_FILE for read: $!";
```

```
# for every line in log file
while(my $line = <LF>){
   # for every keyword
   foreach(@words){
      # save it into $word variable
      $word = $_;
      # if current keyword is found in current line
      if ($line =~ /$word/i){
         # get data groups from line
         # $1 - date
         # $2 - host
         # $3 - reporting process
         # $4 - data
         $line =~ m/(\w{3}\s\d{1,2}\s\d{2}:\d{2}:\d{2})\s(\w+)\s(.+):\s(.+)/;
         # write formatted line to the output file
         print OF "**".$word."**".' '.join(' ', ($2, $1, $4, $3))."\n";
      }
   }
}

# close log and output files
close LF or die "Cannot close $LOG_FILE: $!";
close OF or die "Cannot close $OUT_FILE: $!";
```

```
#!/usr/bin/perl -w

# define variables for naming indexes in array
my $START_DATE = 0;
my $NUM_MERGED = 1;
my $END_DATE = 2;

# define variables with name of files
my $IN_FILE = "output-inter.txt";
my $OUT_FILE = "output-final.txt";

# open input file
open(INF, "< $IN_FILE") or die "Can't open $IN_FILE for read: $!";

# declare array
my @lines;

# read all lines to @lines array
while(<INF>){
   push(@lines, $_);
}

# create new hash
my %new_line = ();

# for every line
foreach(@lines){
        # get date from line
        m/(\w{3}\s\d{1,2}\s\d{2}:\d{2}:\d{2})/i;
```

```
    # save it to variable
            my $date = $1;
            # save line to variable
    my $line = $_;
            # replace date with empty string in the line
    $line =~ s/$date//i;

    # if line exists in the hash
    if (exists($new_line{$line})){
                    # increase number of occurrences
        $new_line{$line}[$NUM_MERGED] = $new_line{$line}[$NUM_MERGED] + 1;
                    # save end date
        $new_line{$line}[$END_DATE] = $date;
    } # else create it and init it with array of next values: start date, number of occurrences, end date
    else{
        push(@{$new_line{$line}}, $date);
        push(@{$new_line{$line}}, 1);
        push(@{$new_line{$line}}, "");
    }
}
# close input file
close INF or die "Cannot close $IN_FILE: $!";
# clear array
undef(@lines);

# for every line in hash
foreach my $str ( keys %new_line ){
            # group string into tokens
    $str =~ m/(.+)\s(\w+)\s\s(.+)/i;
```

```
    $intro = $1;
    $host = $2;
    $data = $3;
            # start to create new line
    $line = $intro." ".$host." ";
    # get array with values for current line
    @values = @{$new_line{$str}};

            # if string was read only once
    if ($values[$NUM_MERGED] == 1){
                    # format string as was
        $line = $line.$values[$START_DATE]." ".$data."\n";
    } else {
                    # else add number of merged lines and end date
        $line = $line."#".$values[$NUM_MERGED]." ".$values[$START_DATE]."-".$values[$END_DATE]."
".$data."\n";
    }
    # add new line to array
    push(@lines, $line);
};

# bubble sort

# for every line
for(@lines){
            # for every line except last one
            for(0..$#lines-1){
                    # if current line greater than next line
                    if($lines[$_] gt $lines[$_+1]){
```

```
                    # swap them
                    ($lines[$_], $lines[$_+1]) = ($lines[$_+1],$lines[$_]);
                }
            }
}

# open output file
open(OF, "> $OUT_FILE") or die "Can't open $OUT_FILE for read: $!";
# save sorted lines to file
foreach(@lines){
   print OF $_;
}
close OF or die "Cannot close $OUT_FILE: $!";
```