```
#!/usr/bin/perl -w
# System Format = Win32
#
######################################################################
##  Technical Services Scripty Thing
##  ================================
##  Author:  Xxx Xxxxxx - Technical Support Officer
##  Creation Date:  Friday, 4th August 2000.
##
##  This script is designed to better manage MARC records that
##  need to be sent to customers and a summary file to NLA.
##  Archive attributes will play a large role in the script,
##  in the future, it is hoped that the ability to automatically
##  send files via FTP to NLA and the DA FTP server for customers.
##
#######################################################################

# Variable initialise
$InputDIR1 = "./wuexport/";
$InputDIR2 = "./cuexport/";
$OutputDIR = "./";
$LastOut   = 0;

# Find the last entrman mkdiry number for output dir
$LastOut = &FindLastOut;
$NextOut = $LastOut + 1;
$NextOut = &GetPadString($NextOut);

# Find the files that need to be outputted in each dir
&GrabFileList;

&CreateInput(1);
&CreateInput(2);

&CopyFiles;

# &WriteOutput;


sub CopyFiles {
    @CopyList = @FileList1;
    foreach $CopyList (@CopyList) {
           next if ($CopyList =~ /^\./);
           next if !($CopyList =~ (/[0-9]/));
           next if !($CopyList =~ (/txt/));
   $x1 = $InputDIR1.$CopyList;
   $x2 = $InputDIR1.$NextOut."/".$CopyList;
   $Tmp = ‘cp $x1 $x2‘;
   print $x1,"\-\>",$x2,"\n";
    }
    @CopyList = @FileList2;
    foreach $CopyList (@CopyList) {
           next if ($CopyList =~ /^\./);
           next if !($CopyList =~ (/[0-9]/));
           next if !($CopyList =~ (/txt/));
   $x1 = $InputDIR1.$CopyList;
   $x2 = $InputDIR1.$NextOut."/".$CopyList;
   $Tmp = ‘cp $x1 $x2‘;
   print $x1,"\-\>",$x2,"\n";
    }
}
```

```

sub CreateInput {
    my $What = shift;
    if ($What == 1) {
   print "Creating DIR: ",$InputDIR1.$NextOut,"\n";
   $I = mkdir ($InputDIR1.$NextOut,0777);
    } else {
   print "Creating DIR: ",$InputDIR2.$NextOut,"\n";
   $I = mkdir ($InputDIR2.$NextOut,0777);
    }
    if ($I) {
        print "Success! \n";
    } else {
        print "Fail! : $! \n";
    }
}

#sub WriteOutput {
#    open (OUTPUT,">$NextOut\.txt") or die ("ERROR opening output file: $!");
#    print OUTPUT "$InputDIR1\n\n";
#    print @FileList1[0];
#    foreach $FileList1 (@FileList1){
#  print "\"",$FileList1,"\"";
#          next if ($FileList1 =~ /^\./);
#          next if !($FileList1 =~ (/[0-9]/));
#  print OUTPUT "$FileList1\n";
#    }
#    print OUTPUT "$InputDIR2\n\n";
#    foreach $FileList2 (@FileList2){
#          next if ($FileList2 =~ /^\./);
#          next if !($FileList2 =~ (/[0-9]/));
#  print OUTPUT "$FileList2\n";
#    }
#}

sub FindLastOut {
   opendir (FINDLASTOUT_OUT,$OutputDIR);
   my @Files = readdir (FINDLASTOUT_OUT);
   closedir (FINDLASTOUT_OUT);
   my $Highest = $LastOut;
   foreach $File (@Files) {
           next if ($File =~ /^\./);
           next if !($File =~ (/[0-9]/));
           if (int(substr($File,0,5)) >= $Highest) {
                   $Highest = int(substr($File,0,5));
           }
   }
   print "Last Entry = ",$Highest,"\n";
   return $Highest;
}

sub GrabFileList {
   opendir (FILELISTDIR1,$InputDIR1);
   @FileList1 = readdir (FILELISTDIR1);
   closedir (FILELISTDIR1);
   opendir (FILELISTDIR2,$InputDIR2);
   @FileList2 = readdir (FILELISTDIR2);
   closedir (FILELISTDIR2);
}

sub GetPadString {
   my $Integer = shift;
   if ($Integer < 10) {
           return "0000".$Integer;
   } elsif (($Integer >= 10) and ($Integer < 100)) {
```

```
                return "000".$Integer;
        } elsif (($Integer >= 100) and ($Integer < 1000)) {
                return "00".$Integer;
        } elsif (($Integer >= 1000) and ($Integer < 10000)) {
                return "0".$Integer;
        } elsif ($Integer >= 1000) {
                return $Integer;
        }
    }



```

```
#!/usr/bin/perl -w
# System Format = Win32
#
###################################################################
##  Technical Services Scripty Thing
##  ================================
##  Author:  Xxx Xxxxxx - Technical Support Officer
##  Creation Date:  Friday, 4th August 2000.
##
##  This script is designed to better manage MARC records that
##  need to be sent to customers and a summary file to NLA.
##  Archive attributes will play a large role in the script,
##  in the future, it is hoped that the ability to automatically
##  send files via FTP to NLA and the DA FTP server for customers.
##
####################################################################

# Variable initialise
@InputDIR       = ("./wuexport", "./cuexport");

# Find the last entrman mkdiry number for output dir
$NextOut = sprintf("%05d", FindLastOut('.') + 1);

# Find the files that need to be outputted in each dir

for my $dir (@InputDir) {
  mkdir "$dir/$NextOut", 0777 or die "Couldn't make dir "$dir/$Nextout": $!";
  CopyFiles($dir, "$dir/$NextOut", GrabFileList($dir));
}

sub CopyFiles {
  my $src = shift;
  my $dst = shift;
  foreach $file (@_) {
    next if $file !~ /\d/;
    next if $file !~ /\.txt$/;
    my $command = "cp $src/$file $dst/$file";
    system($command);
    print "$command\n";
  }
}

sub FindLastOut {
  my $OutputDIR = shift;
  opendir (FINDLASTOUT_OUT,$OutputDIR);
  my @Files = readdir (FINDLASTOUT_OUT);
  closedir (FINDLASTOUT_OUT);
  my $Highest = 0;
  foreach my $File (@Files) {
    next if $File !~ /^\d{5}/;
    my $n = substr($File, 0, 5);
    if ($n > $Highest) {
      $Highest = $n;
    }
  }
  return $Highest;
}

sub GrabFileList {
  my $dir = shift;
  opendir FILELISTDIR, $dir;
  my @files = grep { $_ ne '.' && $_ ne '..' } readdir FILELISTDIR;
  closedir FILELISTDIR;
  return @files;
}
```

```
# This  script  will copy the whole  source directory  tree
# to the  destination. Be carefull  that if the root target
# ( destination ) directory does not exists then it will be
# created. This script will work only on Windows platforms,
# but with some  small changes  will work on unix  as well.
# The destination files will be over  writen if they exist.
# Xxxxxx Xxxxxx
# xxxx@xxxxxxxx.xxxxx.xxx

my $source_directory = ’f:/temp’ ;
my $target_directory = ’c:/temp’ ;

&xcopy($source_directory,$target_directory);

############# xcopy subroutine #############

sub xcopy {
my ($pwd,$i)=($_[0],$i++);
die "You have not defined the \$target_directory variable, sorry...\n" if    $target_directory eq "";
die "You have not defined the \$source_directory variable, sorry...\n" if −d $source_directory !=1;
 if (( −d $target_directory !=1 ) && ( −e $target_directory ==1 ))
 {die "Can’t continue because a file has target’s dir name\n";}
 elsif ( −e $target_directory !=1 )
 { mkdir $target_directory || die "Couldn’t create dir : $target_node\n"}
opendir ($i,$pwd) || die "Can’t list $pwd\n";
while (my $source_node=readdir $i) {
 next if $source_node=~/^\.*$/;
$source_node       = $pwd.’/’.$source_node;
( my $relative_node ) = $source_node =~/$source_directory(.*)/;
$target_node       = $target_directory.$relative_node;
if (−d $source_node==1) {
 if (( −d $target_node !=1 ) && ( −e $target_node ==1 )) {
 die "Can’t mkdir $target_node because a same name file exist\n" }
 elsif ( −e $target_node !=1 ) {
 print "mkdir  $target_node\n";
 mkdir $target_node || die "Couldn’t create dir : $target_node\n" } }
else {
 if (( −d $target_node ==1 ) && ( −e $target_node ==1 )) {
 warn "Can’t copy $target_node because a same name dir exist\n"; }
 else {
 print "coping $source_node to $target_node\n" if −e $target_node !=1 ;
 (my $copy_target_node) = $target_node; $copy_target_node=~s/\//\\/g  ;
 (my $copy_source_node) = $source_node; $copy_source_node=~s/\//\\/g  ;
 ‘$ENV{ComSpec} /c copy /b $copy_source_node $copy_target_node‘;
  }
}
&xcopy($source_node) if −d $source_node==1 }
closedir $i}


```

```perl
# This  script  will copy the whole  source directory  tree
# to the  destination. Be carefull  that if the root target
# ( destination ) directory does not exists then it will be
# created. This script will work only on Windows platforms,
# but with some  small changes  will work on unix  as well.
# The destination files will be over  writen if they exist.
# Xxxxxx Xxxxxx
# xxxx@xxxxxxxx.xxxxx.xxx

my $source_directory = 'f:/temp' ;
my $target_directory = 'c:/temp' ;

&xcopy($source_directory,$target_directory);

############# xcopy subroutine #############

sub xcopy {
  my ($pwd,$i)=($_[0],$i++);
  die "You have not defined the \$target_directory variable, sorry...\n"
    unless -e $target_directory ;
  die "You have not defined the \$source_directory variable, sorry...\n"
    unless -d $source_directory;
  if (! -d $target_directory && -e $target_directory) {
    die "Can't continue because a file has target's dir name\n";
  } elsif ( ! -e $target_directory ) {
    mkdir $target_directory || die "Couldn't create dir : $target_node\n";
  }
  opendir ($i,$pwd) || die "Can't list $pwd\n";
  while (my $source_node=readdir $i) {
    next if $source_node=~/^\.*$/;
    $source_node       = $pwd.'/'.$source_node;
    ( my $relative_node ) = $source_node =~/$source_directory(.*)/;
    $target_node       = $target_directory.$relative_node;
    if (-d $source_node) {
      if (! -d $target_node && -e $target_node ) {
        die "Can't mkdir $target_node because a same name file exist\n";
      } elsif ( ! -e $target_node ) {
        print "mkdir  $target_node\n";
        mkdir $target_node || die "Couldn't create dir : $target_node\n";
      }
    } else {
      if ( -d $target_node && -e $target_node) {
        warn "Can't copy $target_node because a same name dir exist\n";
      } else {
        print "coping $source_node to $target_node\n" unless -e $target_node;
        (my $copy_target_node) = $target_node; $copy_target_node=~s/\//\\/g  ;
        (my $copy_source_node) = $source_node; $copy_source_node=~s/\//\\/g  ;
        `$ENV{ComSpec} /c copy /b $copy_source_node $copy_target_node`;
      }
    }
    &xcopy($source_node) if -d $source_node;
  }
  closedir $i;
}
```

```
# This  script  will copy the whole  source directory  tree
# to the  destination. Be carefull  that if the root target
# ( destination ) directory does not exists then it will be
# created. This script will work only on Windows platforms,
# but with some  small changes  will work on unix  as well.
# The destination files will be over  writen if they exist.
# Xxxxxx Xxxxxx
# xxxx@xxxxxxxx.xxxxx.xxx

my $source_directory = 'f:/temp' ;
my $target_directory = 'c:/temp' ;

&xcopy($source_directory,$target_directory);

############# xcopy subroutine #############

use IO::Dir;

sub xcopy {
  my ($src,$dst) = @_;
  unless (@_ == 2) { die('Usage: xcopy($src, $dst)') }

  my $dh = IO::Dir->new;

  if (-e $dst) {
    die "File $dst exists but is not a directory"  unless -d _;
  } else {
    mkdir $dst or die "Couldn't create dir $dst: $!";
  }
  opendir ($dh,$src) || die "Can't open dir $src: $!\n";
  while (my $file = readdir $dh) {
    next if $file eq '.' || $file eq '..';
    my ($src_file, $dst_file) = ("$src/$file", "$dst/$file");
    if (-d $src_file) {
      xcopy($src_file, $dst_file);
    } elsif (! -e $dst_file) {
      print "coping $file to $target_node\n";
      tr{/}{\\} for $src_file, $dst_file;
      system("$ENV{ComSpec} /c copy /b $src_file $dst_file");
    }
  }
}
```