

```
1    #!/usr/bin/perl -w
2    # System Format = Win32
3    #
4    #####
####
5    ## Technical Services Scripty Thing
6    ## =====
7    ## Author:   Xxx Xxxxxx - Technical Support Officer
8    ## Creation Date:  Friday, 4th August 2000.
9    ##
10   ## This script is designed to better manage MARC records that
11   ## need to be sent to customers and a summary file to NLA.
12   ## Archive attributes will play a large role in the script,
13   ## in the future, it is hoped that the ability to automatical
ly
14   ## send files via FTP to NLA and the DA FTP server for custom
ers.
15   ##
16   #####
#####
17
18   # Variable initialise
19   $InputDIR1 = "./wuexport/";
20   $InputDIR2 = "./cuexport/";
21   $OutputDIR = "./";
22   $LastOut   = 0;
23
24   # Find the last entrman mkdiry number for output dir
25   $LastOut = &FindLastOut;
26   $NextOut = $LastOut + 1;
27   $NextOut = &GetPadString($NextOut);
28
29   # Find the files that need to be outputted in each dir
30   &GrabFileList;
31
32   &CreateInput(1);
33   &CreateInput(2);
34
35   &CopyFiles;
36
37   # &WriteOutput;
38
39
40   sub CopyFiles {
41       @CopyList = @FileList1;
42       foreach $CopyList (@CopyList) {
43           next if ($CopyList =~ /^\.\/);
44           next if !($CopyList =~ ([0-9]/));
45           next if !($CopyList =~ (/txt/));
46           $x1 = $InputDIR1.$CopyList;
47           $x2 = $InputDIR1.$NextOut."/".$CopyList;
48           $Tmp = `cp $x1 $x2`;
49           print $x1,"-\>", $x2, "\n";
50       }
51       @CopyList = @FileList2;
52       foreach $CopyList (@CopyList) {
53           next if ($CopyList =~ /^\.\/);
54           next if !($CopyList =~ ([0-9]/));
55           next if !($CopyList =~ (/txt/));
56           $x1 = $InputDIR1.$CopyList;
57           $x2 = $InputDIR1.$NextOut."/".$CopyList;
58           $Tmp = `cp $x1 $x2`;
59           print $x1,"-\>", $x2, "\n";
60       }
61   }
```

```

62
63     sub CreateInput {
64         my $What = shift;
65         if ($What == 1) {
66             print "Creating DIR: ", $InputDIR1.$NextOut, "\n";
67             $I = mkdir ($InputDIR1.$NextOut, 0777);
68         } else {
69             print "Creating DIR: ", $InputDIR2.$NextOut, "\n";
70             $I = mkdir ($InputDIR2.$NextOut, 0777);
71         }
72         if ($I) {
73             print "Success! \n";
74         } else {
75             print "Fail! : $! \n";
76         }
77     }
78
79     #sub WriteOutput {
80     #     open (OUTPUT, ">$NextOut\.txt") or die ("ERROR opening out
put file: $!");
81     #     print OUTPUT "$InputDIR1\n\n";
82     #     print @FileList1[0];
83     #     foreach $FileList1 (@FileList1){
84     #     print "\"", $FileList1, "\"";
85     #         next if ($FileList1 =~ /^\.\/);
86     #         next if !($FileList1 =~ ([0-9]/));
87     #     print OUTPUT "$FileList1\n";
88     #     }
89     #     print OUTPUT "$InputDIR2\n\n";
90     #     foreach $FileList2 (@FileList2){
91     #         next if ($FileList2 =~ /^\.\/);
92     #         next if !($FileList2 =~ ([0-9]/));
93     #     print OUTPUT "$FileList2\n";
94     #     }
95     # }
96
97     sub FindLastOut {
98         opendir (FINDLASTOUT_OUT, $OutputDIR);
99         my @Files = readdir (FINDLASTOUT_OUT);
100        closedir (FINDLASTOUT_OUT);
101        my $Highest = $LastOut;
102        foreach $File (@Files) {
103            next if ($File =~ /^\.\/);
104            next if !($File =~ ([0-9]/));
105            if (int(substr($File, 0, 5)) >= $Highest) {
106                $Highest = int(substr($File, 0, 5));
107            }
108        }
109        print "Last Entry = ", $Highest, "\n";
110        return $Highest;
111    }
112
113    sub GrabFileList {
114        opendir (FILELISTDIR1, $InputDIR1);
115        @FileList1 = readdir (FILELISTDIR1);
116        closedir (FILELISTDIR1);
117        opendir (FILELISTDIR2, $InputDIR2);
118        @FileList2 = readdir (FILELISTDIR2);
119        closedir (FILELISTDIR2);
120    }
121
122    sub GetPadString {
123        my $Integer = shift;
124        if ($Integer < 10) {
125            return "0000".$Integer;
126        } elsif (($Integer >= 10) and ($Integer < 100)) {

```

```
127         return "000".$Integer;
128     } elsif (($Integer >= 100) and ($Integer < 1000)) {
129         return "00".$Integer;
130     } elsif (($Integer >= 1000) and ($Integer < 10000)) {
131         return "0".$Integer;
132     } elsif ($Integer >= 1000) {
133         return $Integer;
134     }
135 }
136
137
138
```

```
#!/usr/bin/perl -w
# System Format = Win32
#
#####
## Technical Services Scriptor Thing
## =====
## Author: Xxx Xxxxxxx - 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;
}
```

```

1  # This script will copy the whole source directory tree
2  # to the destination. Be carefull that if the root target
3  # ( destination ) directory does not exists then it will be
4  # created. This script will work only on Windows platforms,
5  # but with some small changes will work on unix as well.
6  # The destination files will be over writen if they exist.
7  # XXXXXX XXXXXX
8  # xxxx@xxxxxxxxx.xxxxx.xxx
9
10 my $source_directory = 'f:/temp' ;
11 my $target_directory = 'c:/temp' ;
12
13 &xcopy($source_directory,$target_directory);
14
15 ##### xcopy subroutine #####
16
17 sub xcopy {
18 my ($pwd,$i)=$( $_[0],$i++);
19 die "You have not defined the \ $target_directory variable, sorry...
\n" if $target_directory eq "";
\n" if -d $source_directory !=1;
21 if ( ( -d $target_directory !=1 ) && ( -e $target_directory ==1 ) )
22 {die "Can't continue because a file has target's dir name\n";}
23 elsif ( -e $target_directory !=1 )
24 { mkdir $target_directory || die "Couldn't create dir : $target_no
de\n"}
25 opendir ($i,$pwd) || die "Can't list $pwd\n";
26 while (my $source_node=readdir $i) {
27 next if $source_node=~/^\.*/;
28 $source_node = $pwd.'/'.$source_node;
29 ( my $relative_node ) = $source_node =~/$source_directory(.*)/;
30 $target_node = $target_directory.$relative_node;
31 if (-d $source_node==1) {
32 if ( ( -d $target_node !=1 ) && ( -e $target_node ==1 ) ) {
33 die "Can't mkdir $target_node because a same name file exist\n" }
34 elsif ( -e $target_node !=1 ) {
35 print "mkdir $target_node\n";
36 mkdir $target_node || die "Couldn't create dir : $target_node\n" }
}
37 else {
38 if ( ( -d $target_node ==1 ) && ( -e $target_node ==1 ) ) {
39 warn "Can't copy $target_node because a same name dir exist\n"; }
40 else {
41 print "coping $source_node to $target_node\n" if -e $target_node !
=1 ;
42 (my $copy_target_node) = $target_node; $copy_target_node=~s/\\/\\/\\/;
g ;
43 (my $copy_source_node) = $source_node; $copy_source_node=~s/\\/\\/\\/;
g ;
44 ` $ENV{ComSpec} /c copy /b $copy_source_node $copy_target_node `;
}
45 }
46 &xcopy($source_node) if -d $source_node==1 }
47 closedir $i}
48
49

```

```

1   # This script will copy the whole source directory tree
2   # to the destination. Be carefull that if the root target
3   # ( destination ) directory does not exists then it will be
4   # created. This script will work only on Windows platforms,
5   # but with some small changes will work on unix as well.
6   # The destination files will be over written if they exist.
7   # XXXXXXX XXXXXXX
8   # xxxxx@xxxxxxxxxxx.xxxxx.xxx
9
10  my $source_directory = 'f:/temp' ;
11  my $target_directory = 'c:/temp' ;
12
13  &xcopy($source_directory,$target_directory);
14
15  ##### xcopy subroutine #####
16
17  sub xcopy {
18      my ($pwd,$i)=$( $_[0],$i++);
19      die "You have not defined the \$target_directory variable, sorry...\n"
20
21      unless -e $target_directory ;
22      die "You have not defined the \$source_directory variable, sorry...\n"
23
24      unless -d $source_directory;
25      if (! -d $target_directory && -e $target_directory) {
26          die "Can't continue because a file has target's dir name\n";
27      } elsif ( ! -e $target_directory ) {
28          mkdir $target_directory || die "Couldn't create dir : $target_node\n"
29      };
30
31      opendir ($i,$pwd) || die "Can't list $pwd\n";
32      while (my $source_node=readdir $i) {
33          next if $source_node=~/\^\.*/;
34          $source_node = $pwd.'/'.$source_node;
35          ( my $relative_node ) = $source_node =~/$source_directory(.*)/;
36          $target_node = $target_directory.$relative_node;
37          if (-d $source_node) {
38              if (! -d $target_node && -e $target_node ) {
39                  die "Can't mkdir $target_node because a same name file exist\n";
40              } elsif ( ! -e $target_node ) {
41                  print "mkdir $target_node\n";
42                  mkdir $target_node || die "Couldn't create dir : $target_node\n"
43              };
44          } else {
45              if ( -d $target_node && -e $target_node) {
46                  warn "Can't copy $target_node because a same name dir exist\n";
47              } else {
48                  print "copying $source_node to $target_node\n" unless -e $target_
49                  node;
50                  (my $copy_target_node) = $target_node; $copy_target_node=~s/\/\/\
51                  \\/g ;
52                  (my $copy_source_node) = $source_node; $copy_source_node=~s/\/\/\
53                  \\/g ;
54                  '$ENV{ComSpec} /c copy /b $copy_source_node $copy_target_node';
55              }
56          }
57          &xcopy($source_node) if -d $source_node;
58      }
59      closedir $i;
60  }

```

```
# 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 written if they exist.
# XXXXXXX XXXXXXX
# xxxx@xxxxxxxxxxx.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 "copying $file to $target_node\n";
            tr{/}{\\} for $src_file, $dst_file;
            system("$ENV{ComSpec} /c copy /b $src_file $dst_file");
        }
    }
}
```