Subject: Re: Reading MRI images from GE Signa 3.x/4.x 9-track mag tapes
Posted by David Foster on Tue, 31 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Original Task:

Read MRI image files from GE Signa 3.x/4.x 9-track 1/2"
magnetic tapes, extracting files to disk using subject
name, series number and image number to construct filenames.

Platform: SunOS 4.1.4 on Sparc5, HP 88780 1/2" SCSI tape drive,
Perl 5.004

I got such great help that I thought I would share what I learned, and provide the Perl script that I wrote to take care of this.

The basic method used is a mixture of "dd" and "mt fsf" commands, called using the system() Perl function.

There were 3 ingredients essential to solving this:

1. Use the correct density! Under SunOS the tape device-name specifies the density; use /dev/nrst(1-8) for 800 BPS and /dev/nrst(9-15) for 1600 BPS. For these tapes I needed to use 1600 BPS; I was originally using nrst3 which of course was hopeless.

2. Use the correct block size! Dave Clunie's web site on Medical Image Formats helped a lot. The block size for these tapes is supposed to be 8192, but I found that using 32768 works well and is faster. If we run into reliability problems I'll go back to 8192.

3. There is a "logical-end-of-tape" marker after the first file on the tape. Big pain in the ass! Every utility I tried gave an I/O error. Then one day kinda by accident I discovered that if I ran DD and just ignored the I/O error I could get past this mark.

Once these are taken care of the rest is gravy. Just let DD extract each image file, first to a temporary file, then seek into this file and read the patient's name, series number and image number, and rename the file using this information.

Note that with minor modifications this script could be used on Signa 5.x files as well...the tape format is the

same, but the header information has changed, so you'd
just need to change the file offsets for patient-name,
series# and image#.

Dave Clunie's Med Image Format site:
    http://idt.net/~dclunie/medical-image-faq/html/

GE Format Document for Signa 3.x/4.x:
    Document 46-021858 $56 (call 800-558-2040)

Thanks to everyone that responded. I hope this script
is useful to someone out there.

Dave Foster
--

~~~~~~ ~~~~~~~~~~~
 David Foster   National Center for Microscopy and Imaging Research
  dfoster@ucsd.edu   UCSD/Department of Neuroscience
  (858) 534-7968   http://www-ncmir.ucsd.edu/
~~~~~~ ~~~~~~~~~~~

```
#!/usr/local/bin/perl
#
#   EXTRACT_TAPE  8-30-99  DSFoster
#
# Perl script to extract GE Signa 3.x/4.x MR images off of
# 9-track 1/2" magnetic tapes written in the native Data General
# format. Script simply extracts every MR image and saves onto
# disk, with the subject name, series number, and image number
# incorporated into the filename.
#
# Image files saved with filenames similar to:
#
#     SMITH_JOHN_3_001.mr
#
# Note about using system() to call DD and MT:
#   You must pass a string argument, and not a list, since you
#   want the shell to interpret the string and take care of the
#   shell metacharacters for you (eg "2>"). Be aware that /bin/sh
#   is used, so use the appropriate metacharacters.
#
# Note about using rename() function:
#   The rename() function will not work across filesystem
#   boundaries, so the directory location for temporary
#   file which are to be renamed to the final image filename
#   needs to be the current directory.
```

```
#
# Note about tape device:
#   Under SunOS 4.1.x you must use tape device names /dev/nrst(9-15)
#   for tapes having density 1600 BPS (use (0-8) for 800 BPS).
#   Always use "nrst" to specify non-rewinding!
#
# Note about block size:
#   The block size to use with these Signa 3.x/4.x tapes
#   written using the Data General GE console system is: 32768.
#   The actual block size may be different, but this size works
#   and is quite fast.
#
# Note about file offsets for patient information:
#   The offsets were determined using the following UNIX command,
#   where "file.mr" is an extracted MR image file:
#
#       strings -t d -n 3 file.mr
#
#   These are specific to GE Signa 3.x/4.x image-file format
#   Additional positions:
#       Study: 3136, length=5
#       Plane: 4374, length=16
#       Day:   5188, length=2 (int)
#       Month: 5190, length=2 (int)
#       Year:  5192, length=2 (int)
#
#
# Usage:
#
#   extract_tape [-d] [-v]
#
#       -v : verbose mode (echo commands)
#       -d : debug mode (print variables)
#
#
# Modifications:
#
#  8-30-99 DSF Created.
#


#---- LOCAL CONFIGURATION ----------------------------------------

# Device parameters
#
#    Use nrst(9-15) for 1600 BPS Density tapes.
#    Use nrst(1-8) for 800 BPS Density tapes.
```

```
$tape_device = "/dev/nrst11";            # Tape device
$null_device = "/dev/null";              # Null device

# Filename parameters
#   (eg. SMITH_JOHN_3_001.mr: $name_chars=5, $delim="_", $extension=".mr")

$num_chars = 5;                          # Chars of first/last name
$extension = ".mr";                      # Filename extension
$delim = "_";                            # Filename delimiter

# Name of program (used for naming log file and temporary file)

$prog_name = "extract_tape";

 #------------------------------------------------------------ ------


#---- GE TAPE CONFIGURATION ------------------------------------

# Block size for reading tapes using UNIX "dd" command

$block_size  = "32768";

# Offsets of strings into image files, and lengths of strings.
# This is specific to GE Signa 3.x/4.x Data General tapes.

$NamePos   = 3180;
$NameLen   =   32;
$SeriesPos = 5212;
$SeriesLen =    3;
$ImagePos  = 5208;
$ImageLen  =    3;

# List of expected sizes for image files. Set to ( -1 ) if
# you don't want the size of the image files to be checked.
# (eg. @expected_file_sizes = ( 145408, 139504 );  )
# (eg. @expected_file_sizes = ( -1 );  )

@expected_file_sizes = ( 145408 );

 #------------------------------------------------------------ ------



# Process command-line arguments

$verbose_mode = 0;
$debug_mode = 0;
```

```
for ($i = 0; $i <= $#ARGV; $i++) {
 if ($ARGV[$i] eq "-h") {
  print "\n  Usage:  $0 [-v] [-d] \n";
  print "\n     -v : verbose mode (echo commands)";
  print "\n     -d : debug mode (print variables)\n";
  print "\n    Extracts image files into current directory\n";
  exit 0;
 }
 elsif ($ARGV[$i] eq "-v") {           # Echo system() commands
  $verbose_mode = 1;
 }
 elsif ($ARGV[$i] eq "-d") {           # Debug: print variables
  $debug_mode = 1;
 }
}


# Set temporary filename using local directory (since rename()
#  won't work across filesystem boundaries.

$tmp_file = "." . $prog_name . "." . $$;
if ($debug_mode == 1) { print "\tTemporary file: $tmp_file\n"; }


# Open log file after preventing its overwrite

$log_file = $prog_name . ".log";
if ($debug_mode == 1) { print "\tLog file: $log_file\n"; }
if (-e $log_file ) {
 die "\n    Log file already exists: $log_file ... exiting\n";
}

unless (open LOG_HANDLE, ">" . $log_file) {
 die "\n    Error opening log file: $log_file\n";
}

unless (-w $log_file) {
 die "\n    Do not have write permissions for file: $log_file\n";
}
else {
 printf LOG_HANDLE "\nLog for program \"$prog_name\":  ";
 $tm = localtime();
 printf LOG_HANDLE "$tm";
}


# Install atexit-style handler that will upon exit:
```

```
#    * delete temporary file
#    * rewind tape
#    * write final message to log file
#    * close log file handle

END {
 print "\nRewinding tape...";
 $cmd = "mt -f $tape_device rewind";
 if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }
 $status = system( $cmd );
 print "Done\n\n";
 printf LOG_HANDLE "%s", "\n\nFiles processed:    $processed";
 printf LOG_HANDLE "%s",   "\nErrors encountered: $errors\n";
 close LOG_HANDLE;
 unlink($tmp_file);
}

# Set process umask so files have rw-rw-r-- permissions

umask(002);



#####################################################
# Extract tape label, headers and other junk
#####################################################

$msg = "\n\n  Reading tape label and directory...\n";
print "$msg";
printf LOG_HANDLE "%s", $msg;
$cmd = "dd if=$tape_device of=$tmp_file ibs=$block_size 2> $null_device";
if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }
$status = system( $cmd );

if ($status != 0) {
 print "\n    ERROR reading tape:";
 print "\n      Incorrect density? density should be 1600 BPS";
 print "\n      Is tape loaded and the drive Online?";
 die   "\n      Does the window say \"BOT\", \"1600\" and \"Online\"?\n";
}

# Read past tape mark (produces I/O error but gets us past "problem"
#  tape mark!) Note that you *cannot* skip over this using mt.

$msg = "\n  Reading past tape mark...\n";
print $msg;
printf LOG_HANDLE "%s", $msg;
$cmd = "dd if=$tape_device of=$tmp_file ibs=$block_size 2> $null_device";
```

```
if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }
$status = system( $cmd );

if ($status != 0) {
 print "\n      Received expected I/O error...continuing...\n";
}

# Skip two files

$msg =  "\n  Skipping two files marks...\n";
print $msg;
printf LOG_HANDLE "%s", $msg;
$cmd = "mt -f $tape_device fsf 1";
for ($i = 0; $i <= 1; $i++) {
 if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }
 $status = system( $cmd );
 if ($status != 0) {
  $msg = "\n    Error reading tape...ERROR skipping files.\n";
  printf LOG_HANDLE "%s", $msg;
  die $msg;
 }
}

# Read block of 8192 bytes

$msg = "\n  Reading initial header block...\n";
print msg;
printf LOG_HANDLE "%s", $msg;
$cmd = "dd if=$tape_device of=$tmp_file ibs=$block_size 2> $null_device";
if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }
$status = system( $cmd );
if ($status != 0) {
 $msg = "\n    Error reading tape...ERROR reading initial header block.\n";
 printf LOG_HANDLE "%s", $msg;
 die $msg;
}

# Read every image on tape. First read the image, save to a
#  temporary filename, then get patient information from this file
#  using file offsets. Then rename the temporary file to this
#  new filename. If this new filename already exists, adjust
#  the serial number to force it to be unique.

print "\n=========================";
print "\n  Reading image files...";
print "\n=========================\n\n";
printf LOG_HANDLE "%s", "\n=========================";
printf LOG_HANDLE "%s", "\n  Reading image files...";
```

```
printf LOG_HANDLE "%s", "\n=========================\n\n";

# Initialize variables

$errors = 0;
$processed = 0;
$inc = 1;

while ( $status == 0 ) {

 # Unlink temporary file for good measure

 unlink( $tmp_file );

 # Read image from tape and write to temporary file

 $cmd = "dd if=$tape_device of=$tmp_file ibs=$block_size 2> $null_device";
 if ($verbose_mode == 1) { print "\tCMD: $cmd \n"; }

 $status = system( $cmd );
 if ($status != 0) {
  $errors++;
  $msg = "    Error reading tape...ERROR reading image file [$inc]\n";
  print $msg;
  printf LOG_HANDLE "%s", $msg;
 }
 else {

  # Check file existence and check size if specified

  if (! -e $tmp_file) {
   $status = -1;
   $errors++;
   $msg = "    ERROR saving image file to disk: $tmp_file\n";
   print $msg;
   printf LOG_HANDLE "%s", $msg;
  }
  else {
   unless ($expected_file_sizes[0] == -1) {
    $size = (-s $tmp_file);
    @matches = grep { $_ == $size } @expected_file_sizes;
    if ($debug_mode == 1) {
     print "\tSize = $size\n";
     print "\tMatched size = ";
     for ($i = 0; $i <= @matches; $i++) {print "$matches[$i] ";}
     print "\n";
    }
```

```
  if (@matches == 0) {
   $status = -1;
   $errors++;
   print "    ERROR: unexpected file size: $size\n";
   print "       Allowed size(s): ";
   printf LOG_HANDLE "%s", "    ERROR: unexpected file size: $size\n";
   printf LOG_HANDLE "%s", "       Allowed size(s): ";
   for ($i = 0; $i < @expected_file_sizes; $i++) {
    print " $expected_file_sizes[$i]";
    printf LOG_HANDLE "%s", " $expected_file_sizes[$i]";
   }
   print "\n";
   printf LOG_HANDLE "%s", "\n";
  }
 }
}

# Get information from file and generate new filename for image

if ($status == 0) {

 ($outfname, $Name, $Series, $Image ) =
  CreateFileName( $tmp_file, $NamePos, $NameLen,
    $SeriesPos, $SeriesLen, $ImagePos, $ImageLen,
    $delim, $extension, $num_chars );

 if ($outfname eq "") {        ### Read error: try next file
  $errors++;
  print "    ERROR reading image file; cannot generate filename\n";
  print "       Temporary filename: $tmp_file\n";
  printf LOG_HANDLE "%s",
   "    ERROR reading image file; cannot generate filename\n";
  printf LOG_HANDLE "%s",
   "       Temporary filename: $tmp_file\n";
 }
 elsif (-e $outfname) {        ### File exists: abort!
  $errors++;
  print "    File already exists and cannot be overwritten:\n";
  print "         $outfname\n";
  print "    Move or rename files as necessary and restart program.\n";
  printf LOG_HANDLE "%s",
   "    File already exists and cannot be overwritten:\n";
  printf LOG_HANDLE "%s",
   "         $outfname\n";
  printf LOG_HANDLE "%s",
   "    Move or rename files as necessary and restart program.\n";
  exit 1;
 }
```

```
   else {                    ### Rename file

    # Last image file read

    $LastName = $Name;
    $LastSeries = $Series;
    $LastImage = $Image;

    # Rename image file to new name (rename(): 1=success!)

    $status = 1 - rename( $tmp_file, $outfname );
    if ($status == 0) {
     $processed++;
     $msg = "       ==> Image saved: $outfname \n";
     print $msg;
     printf LOG_HANDLE "%s", $msg;
    }
    else {
     $errors++;
     $msg = "    ERROR renaming file \"$tmp_file\" to \"$outfname\"\n";
     print $msg;
     printf LOG_HANDLE "%s", $msg;
    }

    if ($debug_mode == 1) {
     print "\tOutfname = $outfname\n";
     print "\tName =     $Name\n";
     print "\tSeries =   $Series\n";
     print "\tImage =    $Image\n";
    }
   }
  }
 }

 $inc++;      # Next image file
}

print "\n\nFiles processed:    $processed";
print   "\nErrors encountered: $errors\n";

print "\nLog file saved: $log_file\n";

exit 0;


 #------------------------------------------------------------ -------------
# Get patient info from file and generate filename for image file
#
```

```
# Usage:
#
#    ($outname, $Name, $Series, Image) =
#          CreateFileName( $TmpName, $NamePos, $NameLen,
#               $SeriesPos, $SeriesLen, $ImagePos, $ImageLen,
#               $delim, $extension, $num_chars);

sub CreateFileName {

 # Assign arguments

 $TmpName = $_[0];
 $NamePos = $_[1];
 $NameLen = $_[2];
 $SeriesPos = $_[3];
 $SeriesLen = $_[4];
 $ImagePos = $_[5];
 $ImageLen = $_[6];

 $delim = $_[7];             # Filename delimiter
 $extension = $_[8];         # Filename extension
 $num_chars = $_[9];         # Chars of first/last name to use

 # Open the file and put in binary mode

 unless (open IMG_HANDLE, $TmpName) {
  close IMG_HANDLE;
  print "    ** Error opening file: $TmpName\n";
  return ("", "", "", "");
 }
 binmode IMG_HANDLE;         # Binary mode!

 # Read patient name

 unless (seek IMG_HANDLE, $NamePos, 0) {
  close IMG_HANDLE;
  print "    ** Error in seek() in file: $TmpName\n";
  return ("", "", "", "");
 }
 unless (read IMG_HANDLE, $Name, $NameLen) {
  close IMG_HANDLE;
  print "    ** Error in read() in file: $TmpName\n";
  return ("", "", "", "");
 }

 # Read series number

 unless (seek IMG_HANDLE, $SeriesPos, 0) {
```

```
 close IMG_HANDLE;
 print "    ** Error in seek() in file: $TmpName\n";
 return ("", "", "", "");
}
unless (read IMG_HANDLE, $Series, $SeriesLen) {
 close IMG_HANDLE;
 print "    ** Error in read() in file: $TmpName\n";
 return ("", "", "", "");
}

# Read image number

unless (seek IMG_HANDLE, $ImagePos, 0) {
 close IMG_HANDLE;
 print "    ** Error in seek() in file: $TmpName\n";
 return ("", "", "", "");
}
unless (read IMG_HANDLE, $Image, $ImageLen) {
 close IMG_HANDLE;
 print "    ** Error in read() in file: $TmpName\n";
 return ("", "", "", "");
}
close IMG_HANDLE;          # Close the file

# Generate filename from name, series number and image number

@Names = split(/,/ , $Name);             # Parse Last,First


$FullName = substr($Names[0], 0, $num_chars);   # First name
$FullName = (split(/ /, $FullName))[0];         # Remove whitespace

if (@Names >= 2) {
 $FirstName = substr($Names[1], 0, $num_chars);
 $FirstName = (split(/ /, $FirstName))[0];
 $FullName = $FullName . $delim . $FirstName;
}

# $FirstName = substr($Name, $pos+1, length($Name) - ($pos+1));

$outname = $FullName . $delim . $Series . $delim . $Image . $extension;

return ($outname, $FullName, $Series, $Image);
}
```

File Attachments

1) extract_tape, downloaded 84 times