
Generating and Printing Barcodes, Program Documentation

by Mickey Ellinger

The barcode combines the herbarium acronym ((assigned by the Index Herbariorum and usually referred to as the IH acronym) with the accession number; together they constitute the accession_id, the unique identifier for many records in the SMASCH database. In theory, this combination can assign a unique identifier to any accessioned specimen anywhere in the world.

One or a series of barcodes are generated by two Pascal programs that are compiled as DOS *.exe files, barcodes.exe and multi.exe. These programs can run on any DOS machine to which the barcode printer is attached.

On the DOS machine create a directory \barprog and install the two files barcodes.pas and multi.pas. To customize the programs for your site you will need to make two edits in the barcodes.pas program and one in multi.pas. There is a variable herbariumtype in both programs. Edit its list of values to include the acronyms of the herbaria for which you will want to generate labels. Second, in barcodes.pas edit the procedure setupherbs to use the herbariumtype values, the IH acronym(s) and herbarium name(s) you wish. The program will allow you to set up two acronyms; the user will be able to toggle between them.

Recompile the programs using a Pascal compiler to create two .exe files, barcodes.exe and multi.exe, and copy them to a directory named \barcodes.

In the \barcodes directory create three additional files, prep, bar.bat, and multi.ini. The prep file is simply a text file that contains a number and the name of a barcoder. This name will appear on the barcode. The bar.bat file sets the com port and changes to the \barcodes directory. The multi.ini file sets up the herbarium label for generating a series. Edit it to suit you. Sample prep, bar.bat, and multi.inifiles are included.

include these two lines in your autoexec.bat:

```
cd \barcodes
```

```
bar.bat
```

Imaging, Program Documentation

If you choose to capture images of some or all of your accession sheets and associate them with the database for display via a network browser or other application, you will need to convert your captured images to files small enough

to display over the net and the database will need to know the location of the image files. Our solution was to name the image files after the accession id (which we read into our image capture software using a barcode reader to minimize errors). Included are a c shell script that does the conversion, a mapfile, and a sybperl program that tells the database about the location of the files. We run the scripts from cron overnight, since image manipulation uses a lot of resources.

Conversion

The conversion script is called process_images and is a c shell script that can be edited using any text editor and installed in /usr/local/bin. The script calls utility programs from the pbmplus library, which is freeware available by anonymous ftp from ftp.uu.net and other sites, and cjpeg, which converts to jpg format and is also freeware available by ftp. The script expects to run from the directory where the captured images are to be found. It expects images in TIFF format, which can be compressed or uncompressed. It uncompresses the images before converting them, so the filesystem needs to be large enough to allow for this. The script creates three subdirectories - jpg, gif and munged. It then sets a variable o which is the filename without an extension, the paths for the output files (ofile and otnail), the pbmplus programs and the cjpeg program, as well as a mapfile, browse_map.ppm, that contains a colormap for the browse images. Then it sets a variable z that is the name of the image file with a .tif extension.

The script then calls the pbmplus and cjpeg utilities resulting in a .jpg and .gif image for each .tif image. The .jpg images are ~90k and the .gifs about 10k.

Since we store images on several filesystems and in several directories within them, we wrote a super-simple shell script called /usr/local/bin/find_images that goes to each directory where images might be found and calls /usr/local/bin/process_images. We run that script as a cron job that starts after image capture is finished for the day.

Sybperl and Images

We use a perl program, move.pl, to assign a url path to the image files so that they can be accessed by our web browser. The program can be edited using any text editor and installed in /usr/local/lib/perl. It is a sybperl script and has to run from a computer containing the Sybase DB libraries. It creates a log file, sets the printer name, and prints the log file. We include a printed copy of this log file with our tape archive.

Call the program with the name of your dbo and the dbo password as well as the name of the database, e.g.

```
sybperl move.pl -Umelba -Pt0ast -d smaschtest
```

(Note: make sure that the version that goes public doesn't have our defaults in it!!) and that we take out this line.....

The program opens the log file and an output file, then reads each line of the log file and splits the filename and extension. Then it creates a file that will be copied into the database using the bcp bulk copy utility of sybase. It truncates the bulk copy table and copies the namefile into the table. Then it checks to make sure that there is a database record for the file and prints the names of files for which there is no record. It then deletes records from the temporary file so there won't be any orphan files and updates the query_denorm table. This update causes the update trigger to fire, which assigns the url path. It then creates a tarfile with the appropriate directories that correspond to the url paths assigned to the files.

The tarfile is sent to the image server using ftp. Go to the image directory and ftp the file (in binary mode). Then extract it using the tar xvf tarfile command and it will unwind into the appropriate subdirectories. When you are through rm the tarfile.

We run this script from cron early in the morning so that only moving and extracting the tarfile has to be done manually during working hours.

Inserting Links to Images for the Web Browser

To improve web performance we have created a denormalized table containing a subset of accession, annotation and voucher information, as well as a link to the image file. This table is updated nightly to insert new rows that have been entered during the day. We use a stored procedure, `insert_query_denorm`, which runs as a cron job.