USMARC.pm

Table of Contents
USMARC.pm

NAME

MARC::File::USMARC - USMARC-specific file handling

SYNOPSIS

```
use MARC::File::USMARC;

my $file = MARC::File::USMARC->in( $filename );

while ( my $marc = $file->next() ) {
    # Do something
}
$file->close();
undef $file;
```

EXPORT

None.

METHODS

decode($string [, \&filter_func])

Constructor for handling data from a USMARC file. This function takes care of all the tag directory parsing & mangling.

Any warnings or coercions can be checked in the `warnings()` function.

The `$filter_func` is an optional reference to a user-supplied function that determines on a tag-by-tag basis if you want the tag passed to it to be put into the MARC record. The function is passed the tag number and the raw tag data, and must return a boolean. The return of a true value tells MARC::File::USMARC::decode that the tag should get put into the resulting MARC record.

For example, if you only want title and subject tags in your MARC record, try this:

```
sub filter {
    my ($tagno,$tagdata) = @_;

    return ($tagno == 245) || ($tagno >= 600 && $tagno <= 699);
}

my $marc = MARC::File::USMARC->decode( $string, \&filter );
```

Why would you want to do such a thing? The big reason is that creating fields is processor-intensive, and if your program is doing read-only data analysis and needs to be as fast as possible, you can save time by not creating fields that you‘ll be ignoring anyway.

Another possible use is if you‘re only interested in printing certain tags from the record, then you can filter them when you read from disc and not have to delete unwanted tags yourself.

update_leader()

If any changes get made to the MARC record, the first 5 bytes of the leader (the length) will be invalid. This function updates the leader with the correct length of the record as it would be if written out to a file.

_build_tag_directory()

Function for internal use only: Builds the tag directory that gets put in front of the data in a MARC record.

Returns two array references, and two lengths: The tag directory, and the data fields themselves, the length of all data (including the Leader that we expect will be added), and the size of the Leader and tag directory.

encode()

Returns a string of characters suitable for writing out to a USMARC file, including the leader, directory and all the fields.

RELATED MODULES

MARC::Record

TODO

Make some sort of autodispatch so that you don‘t have to explicitly specify the MARC::File::X subclass, sort of like how DBI knows to use DBD::Oracle or DBD::Mysql.

Create a toggle-able option to check inside the field data for end of field characters. Presumably it would be good to have it turned on all the time, but it‘s nice to be able to opt out if you don‘t want to take the performance hit.

LICENSE

This code may be distributed under the same terms as Perl itself.

Please note that these modules are not products of or supported by the employers of the various contributors to the code.

AUTHOR

Andy Lester, `< <andy@petdance.com>`