

Scraps for Emacs

Program Version 0.91b

Bob Newell
Honolulu, Hawai'i

August 14, 2022

Contents

1	So What's a 'Scrap'?	4
2	Setup	4
3	Alternatives to Scraps	5
4	Using Scraps	5
4.1	Creating New Scraps	5
4.2	Searching	6
4.2.1	Showing Everything	6
4.2.2	Initial Searches	6
4.2.3	Search Results	7
4.2.4	Follow-up Searching	7
4.3	Other Commands	7
4.3.1	Renaming Scraps	7
4.3.2	Merging Scraps	8
4.3.3	Tagging Scraps	8
4.3.4	Random Scrap	9
4.3.5	Show Me What You've Got	9
4.4	Exiting Scraps; Navigating	9
4.4.1	Scraps from Browsers and PDFs	10
5	Migration to Scraps	10
5.1	From Evernote	10
5.2	From Onenote	10
5.3	From Keep	10
5.4	Deft	11
5.5	Dropbox	11
6	Syncing	11
6.1	Dropbox	11
6.2	Rsync	11
6.3	Android	11

6.4	Windows, Mac	12
6.5	i-Whatever	12
7	Background; Usage Tips	12
7.1	Origins	12
7.2	The Main Point	12
7.3	A Usage Example	12
7.4	Clipping	13
7.5	Scraps To-Do	13
7.6	Scraps Alerts and Notifications	13
7.7	Doing Projects	14
7.8	Final Notes	14
8	Shortcomings of Emacs Scraps	15
9	Speed of Scraps	16
10	Error Conditions	16
11	Acknowledgements	17
12	Updates	17
13	Contact	17
14	Bug Reports	17
15	The Legal Stuff	17
16	Changelog	18
17	Wishes and Issues	21
17.1	Open Issues	21
17.2	To-do/Wishlist	22

1 So What's a 'Scrap'?

Very briefly, a 'scrap' is a (maybe) short text file containing information of interest. A collection of such Scraps resides in your default scraps directory. Over time, you may come to have thousands of scraps. This software is all about creating and especially retrieving such scraps.

Scraps aims to be a fast-running replacement for much of Deft and some of Evernote, Google Keep, and/or Onenote functionality. It creates and searches text files in a possibly very large collection.

Much more on all of this further down in the manual.

2 Setup

Before installing or reporting a bug please be sure you have the latest version:

<http://www.bobnewell.net/filez/scraps.zip> .

Obviously I assume some Emacs knowledge. If you need help there are excellent tutorials and videos online, and a vibrant Emacs community which can answer every question far better than I ever could.

Put something like this in your startup file. Set `scraps-directory` to wherever you want to put your scraps (the directory should already exist). The keyboard shortcuts are suggestions, but they are very, very helpful.

An important point: *Don't* store your scraps in a directory that `org-mode` looks in to build the `org-mode` agenda. Once you have a large scraps collection, that would cause building the agenda to be overly slow.

```
(require 'scraps)
(global-set-key "\C-csr" 'scraps-regexp-search)
(global-set-key "\C-csd" 'scraps-deadgrep)
(global-set-key "\C-csw" 'scraps-word-search)
(global-set-key "\C-csR" 'scraps-re-regexp-search)
(global-set-key "\C-csW" 'scraps-re-word-search)
(global-set-key "\C-csn" 'scraps-new)
(global-set-key "\C-csb" 'scraps-bookit)
(global-set-key "\C-csg" 'scraps-from-gnus)
(global-set-key "\C-csm" 'scraps-rename)
(global-set-key "\C-csa" 'scraps-all)
;; Modify as needed, note trailing slash:
(setq scraps-directory "/wherever/you/keep/it/")
```

Be sure `scraps.el` or `scraps.elc` (I recommend byte-compiling) are somewhere in your elisp load path. Or, you can do this instead of `'require`:

```
(load-library "/path/to/scraps.elc") .
```

Scraps probably requires Emacs 26+ and `org-mode` version 9+. It might run with older versions of `org-mode` and Emacs but I have not done any testing of that possibility.

An important prerequisite is now `ugrep` which can easily be installed from your Linux package manager, whichever one your distribution uses. `ugrep` should allow for searching in languages that use multibyte characters.

To use the `scraps-deadgrep` feature you must install the Emacs package `deadgrep` and the command-line `ripgrep` package.

3 Alternatives to Scraps

There is an Emacs program for Emacs called Deft which is also designed to accumulate and search a collection of text files. Actually it's a brilliant piece of software. But the stopper for me was that it doesn't seem to scale to very large collections without getting frustratingly bogged down even on fast computers.

Evernote modes for Emacs don't seem to work any longer, and in any case, Evernote has become Big Business, with ever more severe limits on free accounts and fairly recent large price increases on paid accounts. I was mostly okay with a paid account at \$25 a year but I'm not going to pay \$80 a year.

You can get a Onenote account that does much of what Evernote does for free, although I find Onenote pretty close to unusable, and you have to mess with MS.

There's also Google Keep. That's free and good, but it too doesn't scale to very large collections, becoming very slow when loading. In any event I prefer to stay within Emacs when possible.

4 Using Scraps

4.1 Creating New Scraps

There are any number of ways to add content to your collection. One way is to simply create a text file and save it in your default scraps directory. The only caveat is that the directory should just contain text files. Don't put media files, binaries, etc., in there, or you might cause problems.

Scraps of course provides its own methods to create new content, and that's often the best route.

`scraps-new` (`C-csn`) creates a new scrap, prompting for a name. Again, name it carefully. The suffix `.org` will be appended. You may disagree with this idea. But I did it because then web clippings (and other clippings) can have links preserved, which I think is important. The

links get preserved as `org-mode` links. Why not? However, note that *all* files in your scraps directory will be searched for desired content, not just the `.org` files. This keeps things general and allows you to create more or less arbitrary scraps outside of this software, as noted above. (If you wish you can change the `.org` to something else, but I don't recommend it.)

`scraps-from-gnus` (`C-csg`) creates a new scrap from a Gnus article. Your currently active buffer must be the article buffer (not the summary buffer). You will be prompted for a title. In addition to storing the full text (including displayed headers) of the article, an `org-mode` link will be stored to the original article. Note that attachments will *not* be stored in the scrap, and links to any such attachments won't be valid (this needs to be fixed eventually).

`scraps-bookit` (`C-csw`) creates a new scrap from a region in the current buffer, if a region exists, or a whole buffer, if there is no region. This is useful to make a scrap from, say, an `eww` or `w3m` browser buffer. This is a sort of text-mode Emacs equivalent of the Evernote clipper. (I have separately published `org-web-clipper`, which does a similar thing but saves content as an `org-mode` entry in a single collection file, using a capture template.) You are prompted for a name, to which `.org` will be suffixed. Note that web graphics are typically not preserved (this is about text, after all). If you're clipping from a web page, a link back to the page is saved near the head of the scrap. If you're clipping from a file, a link to the file is likewise saved. If you're clipping from a temporary buffer, no link is saved (obviously).

4.2 Searching

Saving scraps is great, but the whole point is finding them days or even years later. Here's how that works.

4.2.1 Showing Everything

While not a search per-se, `scraps-all` (`C-csa`) shows *all* the scraps in your scraps directory. This is very fast and makes use of previews and other scraps facilities.

4.2.2 Initial Searches

There are two ways to search.

`scraps-regexp-search` (`C-csr`) greps your scraps directory and gives you a `direx` buffer with the results. Grepping is exactly as you would expect. It's case dependent and spans multiple lines in a scrap (a recent change).

`scraps-deadgrep` (`C-csd`) is experimental and greps your scraps directory with the `ripgrep` command (which of course must be installed. The output uses the `deadgrep` interface, not the `Scraps` interface.

`scraps-word-search` (`C-csw`) will only find terms that are preceded and followed by a word boundary, which might be the beginning of a line, a space, a period, etc. But Boolean *and* is

allowed here, using the usual `&` (ampersand) character. If you type for instance `fred&suzie` all scraps containing both the words `fred` and `suzie` anywhere in the scrap will be found. Word searches are case **IN**dependent. This just seemed to make sense. Embedded spaces are preserved, but leading and trailing spaces are not. For instance `_fred_` and `fred` are the same but `fred_suzie` will search for `fred_suzie` with exactly one space between the words (the underscore is meant to represent a space in the examples above).

Regexps are possible within word searches, you'll just have to keep your wits about you, and be sure not to include the `&` character in your searches.

4.2.3 Search Results

If scraps are found corresponding to your search, the screen gets split horizontally into two windows. The upper window is the `dired` listing and the bottom window shows the file corresponding to the current line in the `dired` window.

All `dired` commands work as expected, except `<space>` scrolls the preview window down, and `C-<space>` scrolls the preview window up.

(Well, sort of. `Dired` commands that change the contents of the `dired` buffer require a repeated search to regenerate the display, and I don't know if I've caught all the cases. I've fixed the issue so far for the `x`, `R`, `%r` and `%R` commands but nothing else; suggestions welcome.)

Note that searches grep file *content*, not file *names*. But new scraps automatically have the file name inserted as the first line of the file content, mitigating the issue at least somewhat.

4.2.4 Follow-up Searching

`scraps-re-word-search` (`C-csR`) (note the capital letter) and `scraps-re-word-search` (`C-csW`) do a search within previously obtained results. Think of it as drilling down into your search results, successively narrowing them. If you did a previous regexp search, there is nothing to stop you continuing with a word search, or vice-versa. "re" searches can help make sense out of an overwhelming collection of results. Just be careful, as there's no going back, and if you narrow things too much, you'll have to start over again from the top.

4.3 Other Commands

4.3.1 Renaming Scraps

You can rename the current scrap with the command `scraps-rename` (`C-csm`). The scrap will be renamed and so will the buffer. The buffer gets marked as modified. Be careful: it's saved automatically when you do this, and the new name is fixed up to get rid of problematic characters (like `$`), and to have a (single) `.org` extension.

4.3.2 Merging Scraps

You can merge a group of scraps into a new scrap with **scraps-merge** (no shortcut key at the moment). To select the scraps to merge, search and then mark the selected scraps in the dired buffer (with **m** on the dired line containing the filename). After **M-x scraps-merge** you'll be prompted for a new scrap name; the content of the marked scraps will be put into the new scrap (in dired order) and saved, and you'll be taken to the top of the new scrap. Each merged scrap becomes a top-level **org-mode** entry. The old scraps are *not* deleted; that would be a very bad thing to automate. If you wish you can go back to the dired buffer and delete them with dired commands. Be careful!

4.3.3 Tagging Scraps

Adding a tag to a scrap can aid in finding it later. Just put the desired tag somewhere in the first 12 lines of the scrap or scraps in question.

It's best if tags follow a stricter version of the **org-mode** standard.

1. A tag is bounded by a colon on either side.
2. Tags should be of a minimum length of 3 characters (in addition to the colons) and no longer than 32 characters.
3. A tag should contain only letters and numbers, other than the bounding colons. There should be no spaces, dashes, brackets, etc.
4. A Scraps tag should appear in the first 12 lines of the scrap file.

These rules are not enforced; however, tag inventories with **scraps-tag-list** will only show tags which conform to this standard. The rules are more restrictive than, and do not apply to, conventional **org-mode** tags.

Examples of valid tags are **:hnlrail:** or **:First23Entries:**

Examples of invalid tags are **:a1:** (too short); **:a no 1:** or **:(dakine):** (invalid characters); **:abcdefghijklmnopqrstuvwxy1234567:** (too long).

scraps-tag (no shortcut key as yet) adds a user-specified *tag* to a group of scraps. The tag is consistent with **org-mode** tagging formats. To use this feature, in your scraps **dired** buffer (or really any **dired** buffer), mark (with **m**) each scrap or file that you want to tag. When you've done this, give the **scraps-tag** command. You'll be prompted for a tag name. Don't put any colons in the name; that will be taken care of for you. Then, all marked files will have your chosen tag inserted on a line at the top of the file. The marks will not be cleared, as this would delete part of your work trail.

Note that in general you won't use **org-mode** tag searches to later find scraps with a certain tag. This is because it's not recommended that all your scraps be included in your **org-mode**

agenda; if you have a large number of scraps, that would bog down agenda building to an unacceptable degree. Instead, find tags using the regular Scraps search commands.

scraps-tag-list (no shortcut) will create a temporary buffer containing a complete sorted list of all tags in your scraps directory which conform to the above rules. In this temporary buffer, pressing **s** or **RET** will start a Scraps word search for the tag in question (limited to the first 12 lines of each scrap as per Scraps tag convention).

Of course, if you already know what tag you're looking for, you can search directly with **C-c s w** and skip a step. **scraps-tag-list** helps you out when you can't remember exactly how the tag you want was originally written.

Note too that large collections of scraps which contain many web clippings will likely contain text that looks like a tag even though that isn't what you want. Nothing much to do about that except hand-edit to remove such pseudo-tags.

4.3.4 Random Scrap

The command **scraps-random** (no shortcut key) shows a scrap selected at random from your scraps directory. You can use this as a brain jogger or prompt.

4.3.5 Show Me What You've Got

The command **scraps-size** (no shortcut key as yet) will show the size of your scraps directory and the number of scraps you have. The purpose of this is primarily for determining performance in combination with search speed results reported with each search. It also simply satisfies curiosity.

The command **scraps-version** (no shortcut key) does what you might expect, namely, show the current version number of Scraps.

4.4 Exiting Scraps; Navigating

To exit the dired buffer, press **q** as usual.

If you want to view your whole scraps directory, you can just use dired in the normal manner, or if you want to have scraps keybindings you can regexp search for ***** or just a null string (press 'enter' at the string prompt).

When navigating in the created dired buffer, I capture as many vertical cursor motion keys as I could think of and as were reasonable, so that when the cursor moves vertically, the file preview is updated to reflect the file indicated by the cursor position.

This can be a little error prone, and while it seems to work, feedback on issues is welcome.

I deliberately left out things like file finding, incremental search, etc. When the preview doesn't agree with the cursor position, **C-o** will bring things back into alignment.

Note that *previous* preview buffers get closed when a new preview is opened, unless they have been modified, in which case they are preserved.

4.4.1 Scraps from Browsers and PDFs

I’ve already covered a method of bringing in information from the built-in Emacs browsers `w3m` and `eww` with `scraps-bookit`.

The Scraps code archive includes two bash scripts which assist in creating scraps from the Chrome browser and from PDF files. The scripts, `c2s` and `p2s` are self-explanatory. They have a number of software prerequisites, so please read the comments in the scripts carefully.

The scripts are intended for the `bash` shell, and run standalone, outside of Emacs. However you can always call them from within an Emacs eshell or with `M-x !`. You might want to put them somewhere in your command path.

5 Migration to Scraps

What? You’re on some other system? Hopefully you’ve now realized the benefits of Scraps and want to migrate. (Well, I can always hope.)

5.1 From Evernote

If you want to follow in my most splendid footsteps and migrate from Evernote to Scraps, search online for articles on something like “How to Jump Ship from Evernote.” You’ll be best advised to hack together a couple of shell scripts to change filenames, insert keywords, etc. I’ve got a few sample scripts and have published them in my *35 Years of Emacs* pages, which makes it all relatively easy, even if there are a lot of steps.

5.2 From Onenote

If you’re into Onenote and the rest of the MS ecosystem, you’re probably not reading this in the first place ... but just in case, Onenote does have an export facility of sorts. (See MS’s help pages online.) You can only export one notebook at a time and the available export formats are not especially useful so you will have some additional work to do.

5.3 From Keep

If you’re on Google Keep, you also must do some extra work; take a look at `takeout.google.com`. You’ll eventually have a series of HTML files; use one of the many HTML to text utilities to deal with the HTML markup.

5.4 Deft

This one is easy. Just make your default Scraps directory the same as your default Deft directory. If you want to rename everything to have a `.org` suffix, you can easily do so within `dired`.

5.5 Dropbox

I'm not really all that familiar with Dropbox; can anyone help?

6 Syncing

The ability to sync to other devices is an important feature and is provided by many Scraps alternatives. While Scraps doesn't have sync built-in, there are many options.

6.1 Dropbox

If you run Dropbox you can make your scraps directory a Dropbox folder. This of course requires running the Dropbox daemon, but in this use case you're probably already doing that. (There are several other Cloud options that work the same way.)

6.2 Rsync

For syncing to my other Linux machines, I've used both `strongspace.com` (they offer free 1GB accounts, plenty for your scraps) and `rsync.net` (no free accounts but the prices are reasonable). Both of them implement `rsync` very well (and undoubtedly there are other such services). Syncing is a manual operation, which I prefer, as it avoids the problem of an instant commit of something erroneous. Overall this works very well for me across five Linux devices.

6.3 Android

The Dropbox method works more or less automatically with Dropbox for Android.

It is almost certain that you'll be running Emacs and Scraps from within a Termux environment on Android. So `rsync` works fine, just install from a Termux repository. The same is true of `ugrep`.

6.4 Windows, Mac

I don't know about these. I don't have either a Windows or Mac installation. Dropbox runs on these, and probably there are versions of `rsync` and `ugrep`.

6.5 i-Whatever

No idea. I don't have any of those. I'm sure there's Dropbox and other options. If you have a working setup please let me know and I'll try to document it here.

7 Background; Usage Tips

7.1 Origins

Why is it called Scraps? Because there was an MS-DOS program of that name way back when, thanks to Raymond Lowe, which was fantastic for its time. Alas, original Scraps didn't survive Y2K, stored everything in a single fragile binary database, and in any case didn't scale to multi-megabyte collections.

What's this all about? Raymond Lowe's slogan for MS-DOS Scraps was "back up your brain." It was really a great way to look at things.

A great technical paper appeared in 2008 about the 'scraps' concept although it didn't reference Mr. Lowe's work. You can, as of this writing, find a copy here:

<http://people.csail.mit.edu/emax/papers/T0IS-infoscraps-cameraready.pdf>

.

7.2 The Main Point

The point is to dump items of knowledge, which can be anything at all, into a 'scrap' entry. (In his case it was a database entry, for us it's a usually but not always short text file.) Later on, you can do various searches to find that bit of information. Put addresses, recipes, poems, formulas, restaurant reviews, travel itineraries ... whatever you want ... into a scrap, and you should always be able to find it. Add keywords to make it easier to find. Name your scrap in some meaningful way. (Deft and original Scraps didn't require scrap naming. I do. It's an extra step, but it's quick and adds value when doing retrieval.)

7.3 A Usage Example

As a quick example, supposed you've planned a trip to France and you want to save the flight details. You might press `C-c s n` to start a new scrap, and you might enter the title

“Trip to France July 2017”. That title will populate the first line of your new scrap and then you might enter keywords and text something like:

```
Trip to France July 2017
travel Europe vacation itinerary Dogbiscuit hotel
07/04 11pm-6am Air Dogbiscuit 123 JFK-CDG
07/14 7pm-10pm Air Dogbiscuit 321 CDG-JFK
Uber to JFK, Métro to CDG
Hotel: Maison des Cafards, 5 Rue Régrets, Paris 20ième.
```

Notice the list of keywords making this scrap easy to find even if you don’t remember many details.

7.4 Clipping

I’ve extended the scraps idea to allow a form of *clipping*. Of course, you can cut and paste anything you want into your scrap. But the `bookit` function automates this (see the function description above). It also allows you to clip information from web browser buffers, like `w3m` or `eww`. This is a bit remininscent of the Evernote web clipper. Links are preserved, although images are typically not (as you might expect in a text-mode system). This works surprisingly well and is generally very fast. Clip something into a scrap, add a few keywords and a good title, and always be able to find it. To clip from a GUI browser such as Chrome, see the section above that discusses this.

7.5 Scraps To-Do

If you have `org-mode` set up with capture templates (check online references to see how to do this), it’s easy to associate a `TODO` with a scrap. Simply be sure you’re in the buffer for the appropriate scrap. Then activate the `TODO` capture template. On my system that’s `C-c c t`; your keystrokes may vary according to your setup.

You’ll then be put in a buffer with a `TODO` having a link to the scrap file in question. You can modify this as you wish, to add notes, a deadline or due-date, etc. Then just save, typically with `C-c C-c`.

You don’t want a `TODO` keyword within the scrap itself, because if you’ve followed the recommendation to *not* have your scraps collection be searched when building the `org-mode` agenda (due to speed considerations), your `TODO` won’t be found.

7.6 Scraps Alerts and Notifications

To set an alarm, alert, or notification on a given scrap, you need to do a few things. First, create a `TODO` on the scrap, as described just above. Then, make sure there is a schedule or deadline on the scrap. You might perhaps set this with `C-c C-s` while in the `TODO` entry.

Then, use the `org-alert` package to set up your notifications. I won't describe that here; install the package (likely from MELPA) and consult the package documentation.

7.7 Doing Projects

Here's a quick tip for doing projects with Scraps. Now, Scraps is *not* a project management system, but nevertheless you can store all sorts of odds and ends very quickly and with easy retrieval. Simply add a project tag to your scrap. Put it right in the title. For example, if you're working on the Honolulu rail project (poor you!) you might title a scrap like this:

```
Inspect failing foundations in Pearl City :hnlrail:
```

The tag `:hnlrail:` is very likely to be unique and therefore it's easy to find all scraps having to do with this illustrious project.

7.8 Final Notes

The more you put into your scraps collection, the more you get out of it, and there is exponential benefit gain. I have built up thousands of scraps (which is why I needed something that scaled) and I can find stuff from years and years ago. (This collection was built over many years and migrated from system to system. It could use some pruning, but why bother?)

Please note that the Scraps concept is *not* about text mining, finding associations, creating hierarchies, etc. Hence there are no subdirectories, no tree structures, etc., and only a basic tagging capability. Scraps is about storing random information in quantity, and being able to retrieve it later. Want to find that restaurant you went to three years ago that served super great Thai food? If you put that in a scrap, you'll retrieve it, even if you can't remember whether it was in Cincinnati or Tokyo.

Also please note that Scraps is *not* intended to be a general purpose text indexing system. My current collection is about 4,000 scraps occupying 70MB, and Scraps scales very well to perhaps the 100MB level, maybe more. But much as Mr. Lowe's original Scraps didn't scale to the megabyte level, my Scraps for Emacs surely won't scale to the gigabyte level let alone the terabyte level. Again, the idea is small 'scraps' of random information, not large collections of long documents.

That said: I use Scraps to save just about *every* small bit of information. I create new scraps daily, and when I'm doing research into some topic, I'll create lots of them, and worry about organizing and combining them later, if ever. When you realize that if you make a scrap from something, even just some little detail, you'll always be able to retrieve it, then you realize the power of the concept and it becomes a habit or even an obsession. I've at times carried it pretty far, doing things like finding an interesting novel on Gutenberg and saving *the whole thing* into a 3 MB scrap. I don't recommend you do the same (or at least not very often).

I can't take credit. I certainly didn't invent the concept and Emacs Scraps is not the only implementation, nor can I claim it's anything like the best. But I use it. A *lot*.

8 Shortcomings of Emacs Scraps

Scraps isn't perfect, and there are features found in other software that aren't present, at least not yet. Here's a quick and honest rundown.

1. Perhaps most significant of all, Scraps does not seem to work properly with languages using multi-byte characters. Solving this problem doesn't seem easy, and unfortunately I can't promise to get it done. It would be a giant improvement but alas, sufficient time is lacking. I would gladly entertain patches from anyone inclined to look into this.
2. Scraps has no incremental search such as is seen in Deft.
3. Scraps has no inline file preview such as found in Deft, although there is a split-buffer preview of the file under the cursor in the `dired` buffer.
4. Scraps has no auto-save. I don't want a `run-with-timer` process going all the time. It's a tradeoff.
5. Scraps doesn't screen out binary files from preview. Do you really have them in a directory of text files?
6. Scraps functions do not propagate into subdirectories. Do you really have subdirectories in your scraps directory? The whole point is not to have to do that, and adding a subdirectory feature is harder than I originally thought.
7. Scraps doesn't allow for multiple scraps directories, let alone searches across all or some of them; it's just one to a customer right now. This might (or might not) be a desirable improvement; again, it's not a general-purpose indexing tool.
8. Scraps does not generally save graphics or a full web-page view, unlike Evernote, which does.
9. Scraps doesn't allow for voice and/or media notes, such as can be done in several other software options.
10. Compatibility across devices is not automatic, as Scraps doesn't explicitly use cloud storage.
11. There's no searching within PDFs, no OCR, etc. Scraps is based on text (although many files can be converted to text).

But remember that Scraps is free, fast, scalable, and integrated with the Emacs and `org-mode` ecosystems.

9 Speed of Scraps

Search speed depends on the size of the collection (obviously) and whether or not the search string is pure ASCII. With smallish collections of a couple of hundred scraps totaling a megabyte or so, speed really isn't an issue.

It's in large collections, such as accumulate over a period of months and years, that speed becomes a real consideration.

Here are some of my own results.

An initial search with international characters takes a little time on my huge collection, but it's still quite fast enough. Initial search time for a 2,600 item, 56 MB collection (a previous state of my Scraps directory) was about 3 seconds on my i7 desktop. On my i5 SSD equipped laptop, search time was under 4 seconds. On my old old netbook, with a slow single core processor and only 2 gigs of memory, search time was 20 seconds— quite speedy considering the ancient hardware— until I replaced the spinning hard drive with an SSD; that dropped initial search time to under 4 seconds. This verifies that initial searches are I/O bound in large collections.

Now, I say “initial search” because all the scraps have to be read from disk the first time around. There's no getting around this, and as noted above, speed is really defined by how fast your disk subsystem works.

A search without international characters is somewhat faster on an initial search, the exact amount depending mostly on processor speed.

On subsequent searches, when some or all of the scraps have been cached, a search with international characters is typically at least ten times faster; on my i7 desktop, it's about 0.3 seconds. Of course, as you do other things and cache gets replaced, you'll lose some of this speedup.

Without international characters, a second search is substantially faster, but we're talking about improvements to sub-second speeds.

So, what's the story about ASCII vs. international characters? Scraps detects whether the search string contains international characters. If it doesn't, scraps uses a certain speedup hack that makes **grep** scan files at a dramatically faster rate. When those files are cached, the speedup is maximized. However for large collections the limiting factor will always be the speed of the disk subsystem.

10 Error Conditions

I think I've fixed the issues with international characters and special characters in scraps file names, but if anything odd takes place please let me know. For certain, if you get a 'file not found' error during a search, there is a file in your scraps directory with a pathological name and you need to dispose of it by renaming or deleting. Please let me know if this happens

so I can modify the name filter. And it's worth repeating: multibyte characters are still a big unsolved issue.

11 Acknowledgements

Deft, original Scraps, peep-dired, and others provided ideas and techniques in abundance, and my thanks to the authors of all of those. Commercial entities: no, your patents or copyrights haven't been infringed, so don't say they were.

12 Updates

I don't announce updates on the mailing lists, as I don't wish to generate possibly unwanted traffic. The latest version will always be here:

<http://www.bobnewell.net/filez/scraps.zip>

and there is additional information and software on my Emacs web pages, *35 Years of Emacs*:

<http://www.bobnewell.net/publish/35years/index.html>

Eventually Scraps may be on MELPA. But not yet.

13 Contact

That's it for now. Comments, suggestions, use cases, etc. go to scraps@bobnewell.net and are most welcome.

14 Bug Reports

SCRAPS IS FUNCTIONAL BUT IS BETA RELEASE SOFTWARE. TESTING IS INCOMPLETE AND HAS ONLY BEEN ON LINUX SYSTEMS. THERE WILL BE BUGS. Please report them to scraps@bobnewell.net. No idea if this will work on Windows or Mac. Please let me know how it goes if you try either of those. Be sure to try the latest version before reporting bugs.

15 The Legal Stuff

Given the prevalence of sue-ers and patent and copyright trolls, the usual disclaimers must apply.

No support is promised or offered. No liabilities of any kind whatsoever are accepted under any legal theory whatsoever, and I will not be responsible for any damages of any type whatsoever under any legal theory whatsoever even if I am informed of the possibility of such damages. You must use this software at your own risk and make your own evaluation of risk and utility. There are no warranties of any type including implied warranties including but not limited to merchantability or fitness for intended purpose. THIS SOFTWARE MAY CAUSE DATA LOSS AND YOU ACCEPT RESPONSIBILITY FOR ANY SUCH LOSS. IT IS YOUR RESPONSIBILITY TO BACK UP AND SAFEGUARD YOUR DATA.

Versions from 0.10a and later are Copyright (C) 2017-2020 Bob Newell. All rights reserved. You are granted a cost-free, unlimited, permanent license to use this code in any legal manner that you may wish, without restriction, except that you may not exert copyright over, patent, or claim ownership of, the original code or ideas expressed in the code. If you send me suggestions, bug reports, code additions, advice, or anything else of any nature pertaining in any manner to this code, you agree that it becomes my property for me to use as I see fit without cost or obligation of any nature whatsoever, including but not limited to recognition of you as the contributor. If you create derivative work(s) from this code, you agree that this code, or any subsequent variants and releases of this code, regardless of changes in scope or content, do not infringe and have never infringed upon any copyrights or patents you may assert in your derivative work(s).

If you do not accept these terms and conditions you may not use or possess the code.

16 Changelog

This was going to be a two-hour project and initially it was, but it stretched out to days and weeks and months (and now years!), and has provided and continues to provide a wonderful excuse for putting off much more important work. Emacs is like that!

2022-08-14 0.92b Temporary switch back to 'grep' because ugrep
has an issue with spaces in search strings.
2022-04-18 0.90b Switched to 'ugrep' to allow for languages that
use multibyte characters (experimental).
0.91b Added experimental use of 'deadgrep' via
'scraps-deadgrep'.
2021-01-28 0.85b Added scraps-from-gnus to make a scrap directly
from a Gnus article.
2020-10-01 0.84b Added option to interactively reuse existing
scrap when creating new with duplicate name.
Changed to numerical suffixes.
Removed obsolete 'clipping' section from manual.
2020-09-30 0.83b Made '+' suffixes for new scraps optional
when creating interactively.
2020-09-29 0.82b Added 'scraps-all' to show all scraps.

2020-05-14 0.81b Make sure ' is not in scrap name.

2020-05-13 0.80b Backed out potential use of rg as it caused compatibility issues.
Added -z to grep options so that regexp searches can span multiple lines in a file.
Minor doc changes.

2020-02-08 0.77b Added random scrap feature.

2018-04-15 0.76b Backed out advice on link functions as it's now fixed in org itself.
Fixed a malformed dired regexp rename call.

2018-04-13 0.75b Advise org-w3m- and org-eww- link functions to trap error. May be temporary.

2018-02-21 0.74b Fix display in Termux when capturing from w3m or eww.
Resolve a few compile warnings.

2017-11-17 0.73b Eliminate '&' from file names.

2017-09-24 0.72b Refactoring manual. No version change.

2017-09-20 0.72b Fixed incredible stupidity in word-search, and simply use grep's -w option.
Added tag list feature. Surprisingly hard to do in a sensible manner. Updated docs to reflect new feature.
Alarm feature shunted to org-alarm (see manual).
Started to add code for eventual OR searches.

2017-09-19 0.71b Added group tagging feature.
Added procedures on to-do to the manual.
Added version command.
Outlining 'alarm' feature.

2017-09-17 0.70b Incorporated lang=c grep hack when search is pure ascii (with auto-detection of character set), to get a significant speedup when searching large collections.
A few changes to the manual.
Bumped to 0.7 series as new features are in the process of being added.

2017-08-28 0.63b Added scraps-size command.

2017-08-27 0.62b Refactor scrap cleaning with combined regexps.
Insert unmodified scrap name into new scrap rather than inserting 'fixed' name.

2017-08-26 0.61b On a scrap rename, insert the new scrap name at the top of the scrap, to preserve convention.

2017-08-26 0.60b Completed LaTeX documentation. Removed most documentation from the source code.

2017-08-25 0.50b Added a much-needed scraps merge feature.

Bumped version number to recognize this.
 Did some refactoring in the name check code
 to ensure new scrap names are always checked.
 Fixed bug in dired rebuild process, as search
 within search was taking place after a
 scraps-rename operation.

2017-08-24 0.48b Added scraps-rename feature.
 Further cleanup on new scrap name.

2017-08-22 0.47b Fixed bug triggered when a filename contained
 the '\$' symbol in certain positions.

2017-08-17 0.46b Fixed bug in dired delete/rename where previous
 results were incorrectly reused instead of a
 completely fresh re-search.

2017-08-12 0.45b Implemented some scrap name cleanup, which is
 necessary for external web clipping.
 Did a little more initialization.

2017-08-11 0.44b Fixed bug that arises if a subdirectory
 is present in scraps directory. The subdir
 is silently skipped over.
 Added protection against duplicate scrap names.
 Fixed bug where slashes in scrap name cause
 attempt to create subdirectories.

2017-08-11 0.43b Added file-path insertion for non-browser
 buffer extraction in 'scraps-bookit'.
 Added additional material to documentation.

2017-08-10 0.42b Found and fixed major bug with default
 directory being altered in non-scrap buffer.
 Added original URL insertion to scrap created
 from w3m or eww buffer in 'scraps-bookit'.

2017-08-10 0.41b Edited and added to docs.
 I'm confident enough to go back to 'beta'.

2017-08-08 0.40a Added re-search (search within results).
 Refactored search code, much cleaner.

2017-08-06 0.31a Fixed all scraps naming problems.
 Simplified dired invocation.
 Now use default dired buffer names.

2017-08-05 0.30a Major refactoring and changes.
 Split into word search and regexp search.
 Allow boolean & for word search.
 Word search is case-independent, regexp is not.
 find-grep-dired no longer used.
 Removed some now incompatible commands.
 Fixed multibyte-characters-in-filename problem.
 No more asynchronous stuff.

2017-08-04 0.21a Added experimental word/boolean search.
 Not of release quality yet. Back to 'alpha'.

2017-08-04 0.20b Bumped to 'beta' status.
 Put preview scrolling back in.

2017-08-04 0.12a Made 'q' command do some cleanup.

2017-08-03 0.11a Put scraps title into new scrap for easier finding.
 Did some function separation.
 Forced wait when rebuilding find buffer; this is asynchronous just like the original build.

2017-08-03 0.10a Eliminated peep-dired dependency! This was easier than I expected, and let me solve all sorts of problems with file preview changes required by cursor motion.

2017-08-03 0.04a Fixed basic scraps command to be MUCH simpler!
 Can't believe the stupid way I coded it before.
 Changed initial file preview call.
 Partial fix for the find/peep interaction issue.
 Identified peep-dired positioning issue and submitted a patch (for peep-dired) on github.
 Restored error messages on search-forward and -backward.

2017-08-02 0.03a Fixed a text search problem with spaces in input.

2017-08-01 0.02a Added timer.
 Fixed a search bug.

2017-07-31 0.01a Initial coding. Bob Newell, Honolulu, Hawai'i.

17 Wishes and Issues

17.1 Open Issues

- In the Scraps dired buffer, not all motion commands and buffer-change commands are properly dealt with, but hopefully many of them are (although notably, incremental searching in a dired buffer is not). I didn't want to implement the usual `post-command-hook` method of catching cursor motion as it's just a bit on the heavy side.
- There are a lot of global variables that could be local. While this was great for debugging, it's not good style. I'm starting to fix this but it doesn't have (or deserve) much priority.
- There are always bugs. Always. Just when I think I'm done ... another one pops up.
- Scraps should really go into one of the elisp repositories like MELPA. Really, it should,

but that means setting up on github and making a sort of implied support commitment ... some day, perhaps.

17.2 To-do/Wishlist

- Boolean OR is not implemented. AND is by far the most useful operation but there may be times when OR would be useful. I've started to sketch in code for this, but it's much more involved than it at first appeared, and this may take a while. A long while.
- On a re-search, if you narrow things too much, you have to start over completely. There isn't a facility to pop back to previous searches. I might be able to implement this as a side-effect of coding Boolean OR, but it is not a big priority.
- Scraps, when presented in the dired listing, are in alpha order. There is no facility to present them newest first. I thought this would be easy but it isn't. The dired manual section says this: "If DIRNAME is a cons, its first element is taken as the directory name and the rest as an explicit list of files to make directory entries for. In this case, SWITCHES are applied to each of the files separately, and therefore switches that control the order of the files in the produced listing have no effect." In other words, `-t` is an effective no-go. I can think of difficult and slow ways to sort by time, but that isn't what I want.