The horror of chat logs

Can I has normalized, plain text chat logs, please? Starting from 2005, from 7 different clients.

NOTE: this is not a copy-paste code. Special thanks to https://regex101.com.

My chat experience started somewhere in 1998, and it started with a web java IRC client alongside with ICQ 98a. *(For those who haven't seen this, it looked something similar[^1].)*

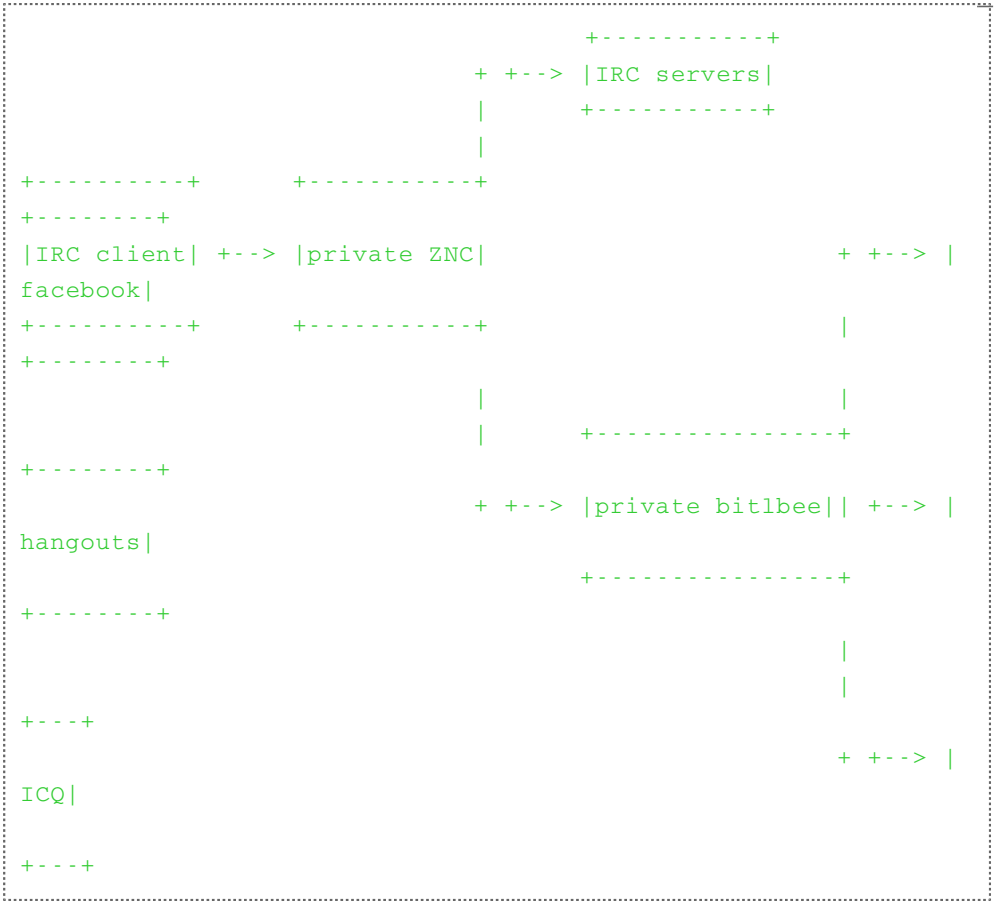
Unfortunately I was too noob to know what logs and backups are, and so far I wasn't able to find any remaining relics from that era at all.

Soon I moved to mIRC instead of the web client, but I'm unable to recover logs from that time either. To be honest, this made me pretty sad; these are memories, just like letters are for previous generations. Being fairly certain I lost them all is not a good feeling.

To prevent this happening in the future I decided to jump into a massive task of 'normalizing' all the logs I still have - that is from ~2005 - and this is when I realized the horror of logging.

My current flow

I've recently moved my communication away from the regular client-server way. Right now I have:



(thanks http://asciiflow.com/ for the ascii "art")

Notes:

- ZNC[^2] is an IRC proxy; you can connect to it with a lot of clients simultaneously and it'll keep you logged in to the services.
- bitlbee[^3] is a little magic server that makes it possible to connect with IRC to services like Facebook, Hangouts, ICQ, and a lot more.

The clients are not logging at all; the only thing that does logging is ZNC, in a consistent, text format: `{znc account}/{znc channel}/{channel or user}/{YY-MM-DD}.log`

and the lines are: `{hh:mm:ss} <{user}> {message}`

I'm aware that this lacks information like exact contact account, but these days both the Facebook and the Gtalk account names are messed up and pretty unusable.

These logs then are synced to other devices via synthing[^4].

The duplicates problem

I've made fairly random, definitely non incremental backups, therefore I have tonnes of duplicates I need to get rid of. Eliminating line duplicates is that nasty *(bash$ cat | sort | uniq)*, but do be able to easily fix things I decided to create a MySQL database as intermediate, temporary storage.

chatlogs.sql

```sql
SET NAMES utf8mb4;

CREATE TABLE `chatlogs` (
  `time` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `medium` varchar(42) NOT NULL,
  `account` varchar(255) NOT NULL,
  `buddy` varchar(255) NOT NULL DEFAULT '',
  `from` varchar(255) NOT NULL DEFAULT '',
  `text` text NOT NULL,
  `hash` varchar(160) CHARACTER SET ascii NOT NULL,
  PRIMARY KEY (`hash`),
  KEY `time` (`time`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

The hash field contains a sha1 hash:

```php
$hash = sha1( $medium . $account . $buddy . $time );
```

so by checking against this, no duplicates will be added.

What lies ahead

During the years I've used many clients, but logs I have only from:

- messenger plus[^5]
- trillian[^6]
- pidgin[^7]
- skype[^8]
- empathy[^9]
- thunderbird[^10]
- hex and xchat[^11] client side logs
- Facebook[^12] online

Obviously all of these used different logging formats.

The common block

Yes, I wrote this in PHP. And yes, I even used globals. This is just a script, not a 'real' program. If you don't like it, write something else ;)

chatlogs_common.php

```php
<?php

global $mysqli;
$mysqli = new mysqli("<server>", "<user>", "<pass>", "<db>");
if ($mysqli->connect_errno) {
    echo "Failed to connect to MySQL: (" . $mysqli->connect_errno . ") " . $mysqli->connect_error;
}

function _common_pre ($str) {

    // trillian entries are urlencoded
    $str = urldecode($str);
    // get rid of any html tag
    $str = strip_tags($str);
    // decode html magic
    $str = htmlspecialchars_decode($str);
    $str = html_entity_decode($str);
    // trim whitespaces
    $str = trim($str);

    return $str;
}

function _common_post ($str) {
    global $mysqli;
    // replace newline with a space
    $str = preg_replace( "/\r|\n/", " ", $str );
    // just in case I missed something
    $str = mysqli_real_escape_string( $mysqli, $str );
    return $str;
}
```

```
function _escape_account ($str) {
    $str = _common_pre($str);
    // this might go into a filename, so make it filename-safe
    $str = preg_replace('((^\.)|\/|(\.$))', '_', $str);
    $str = _common_post($str);
    return $str;
}

function _escape_medium ($str) {
    $str = _common_pre($str);
    // too much variety on upper/lowecase => lowercase all
    $str = strtolower($str);
    // this might go into a filename, so make it filename-safe
    $str = preg_replace('((^\.)|\/|(\.$))', '_', $str);
    $str = _common_post($str);
    return $str;
}

function _escape_buddy ($str) {
    $str = _common_pre($str);

    // you can do magic here, if you need to, these are
examples
    /*
    if (strstr($str,'@conference.jabber.your.server.chat'))
        $str = '#' .
str_replace('@conference.jabber.your.server.chat', '', $str);
    */

    /*
    if (strstr($str, 'Group Conversation '))
        $str = '#' . 'conversation_' . str_replace('Group
Conversation ', '', $str);
    */

    /*
    if (strstr($str, '@conference.jabber.your.server')) {
        $s = split('@', $str);
        $str = str_replace('conference.jabber.your.server/',
'', $s[1]);
```

```
    }
    */

    /*
    if (strstr($str, '@special.domain')) {
        $s = split('@', $str);
        $str = strtolower(str_replace('.', '', $s[0]));
    }
    */

    // this might go into a filename, so make it filename-safe
    $str = preg_replace('((^\.)|\/|(\.$))', '_', $str);

    $str = _common_post($str);
    return $str;
}

function _escape_from ($str) {
    $str = _common_pre($str);
    // pidgin logs can be messy
    $str = rtrim($str, ':');
    // since the end text will contain this within <> I'd like
to keep those files
    // parseable
    $str = str_replace('<', '&lt;', $str);
    $str = str_replace('>', '&gt;', $str);
    $str = _common_post($str);
    return $str;
}

function _escape_text ($str) {
    $str = _common_pre($str);
    // newlines, be <br />
    $str = nl2br($str);
    $str = _common_post($str);
    return $str;
}

function _hash_exist ($hash) {
    global $mysqli;
```

```
    $res = $mysqli->query("SELECT `time` FROM `chatlogs` WHERE
`hash` = '{$hash}';");
    $row = $res->fetch_assoc();
    if (!empty($row))
        return true;

    return false;
}

function _insert ($time, $medium, $account, $buddy, $from,
$text) {
    global $mysqli;

    $medium = _escape_medium($medium);
    $account = _escape_account($account);
    $buddy = _escape_buddy($buddy);
    $from = _escape_from($from);
    $text = _escape_text($text);

    if (empty($medium) || empty($account) || empty($buddy) ||
empty($from) || empty($text) || empty($time))
        return;

    $hash = sha1( $medium . $account . $buddy . $time );
    if (_hash_exist($hash)) {
        return;
    }

    $mysqli->query("INSERT INTO `chatlogs` (
    `time`,
    `medium`,
    `account`,
    `buddy`,
    `from`,
    `text`,
    `hash`
    ) VALUES (
    '{$time}',
    '{$medium}',
    '{$account}',
    '{$buddy}',
```

```
    '{$from}',
    '{$text}',
    '{$hash}')");
}
```

Messenger Plus!

I'm not proud I used Messenger Plus, but at that time, it was a reasonable choice as I was not yet a full fledged nerd.

Logs are in HTML files. Days are separated by <h2> tags and a hidden div, containing a proper timestamp. An <ul> with to <li> elements are storing the from and to account information. The messages are in tables (!). The only positive is that it nicely formatted, so the linebreaks at least I can rely on; the rest is up to the regexes. They are also in UTF-16 and the time is lacking seconds.

The filenames are:

```
/{contact}/{broken charset Hungarian Month longname} {year}.{html}
```

Getting the contact from the filename is doable.

I've put the messenger plus logs into the folder 'messenger-plus' and generated a list of need to be processed files as:

`chatlogs_messenger_todo.sh`

```bash
#!/bin/bash

cd messenger-plus
r=$(pwd)
r=${r//\//\\\/}
find . -iname *.html | sed "s/\.\//$r\//g" > html_todo
```

`chatlogs_messenger.php`

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'messenger-plus' . DIRECTORY_SEPARATOR .
'html_todo' ));

foreach ($files as $file) {
```

```
    if (empty($file))
        continue;

    $fdetails = pathinfo($file);
    $fdetails['dirname'] = str_replace('/absolute/path/to/
messenger-plus/','',$fdetails['dirname']);
    $details = split("/", $fdetails['dirname']);
    $medium = 'msn';
    $buddy = $details[0];

    $lines = split("\n",iconv('utf-16', 'utf-8',
file_get_contents($file)));

    $date = date('Y-m-d');
    $account = '';
    $prawtime = '';
    $psec = 0;

    foreach ($lines as $line ) {
        $datematch = $accountmatch = $contactmatch = array ();

        // look for date split
        if (preg_match_all('/<div class="mplsession"
id="Session_(.*?)">/', $line, $datematch)) {

            if (isset($datematch[1][0])) {
                $t = split("T", $datematch[1][0]);
                $date = $t[0];
            }
        }

        // look for account name
        if (preg_match_all('/<li class="in">(.*?) <span>\((.*?)
\)<\/span><\/li>/', $line, $accountmatch)) {
            if (isset($accountmatch[2][0])) {
                $account = $accountmatch[2][0];
            }
        }

        // look for contact name
```

```
        if (preg_match_all('/<li>(.*?) <span>\((.*?)\)<\/
span><\/li>/', $line, $contactmatch)) {
            if (isset($contactmatch[2][0])) {
                $buddy = $contactmatch[2][0];
            }
        }

        // look for message lines
        if (preg_match_all('/<tr><th><span class="time">\((.*?)
\)<\/span> (.*?):<\/th><td>(.*?)<\/td><\/tr>/', $line,
$messagematch)) {
            $t = $messagematch[1][0];
            $rawtime = $date . ' ' . $t;
            // adding unique to minute seconds
            if ($rawtime != $prawtime) {
                $sec = 0;
            }
            else {
                $sec = $psec + 1;
            }
            $prawtime = $rawtime;
            $psec = $sec;

            $sec = str_pad($sec, 2, '0', STR_PAD_LEFT);
            $time = $rawtime . ':' . $sec;

            $from = $messagematch[2][0];
            $text = $messagematch[3][0];

            _insert($time, $medium, $account, $buddy, $from,
$text);
        }

    }

}
```

Trillian

Trillian stores *(used to store, we're talking ancient versions here, like 0.72)* logs in broken XML: valid line by line, not valid as file, and everything is stored in attributes. There are some plain text files as well, but I've decided to use the XML: it stored unix timestamps, and the text was urlencoded, so I didn't have to deal with unexpected linebreaks. *(This eventually happened when I started dealing with Pidgin, so much for trying to save myself from a nasty regex.)*

Files look something like this:

```
{medium}/{Query,Channel}/{contact,conversation}.xml
```

Getting the medium and the contact information from the filename is doable.

chatlogs_trillian_todo.sh

```bash
#!/bin/bash

cd trillian
r=$(pwd)
r=${r//\//\\\/}
find . -iname *xml | sed "s/\.\//$r\//g" > xml_todo
```

chatlogs_trillian.php

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'trillian' . DIRECTORY_SEPARATOR .
'xml_todo' ));

foreach ($files as $file) {

    if (empty($file))
        continue;
```

```
    // this can also be group conversation name
    $buddy = pathinfo($file, PATHINFO_FILENAME );

    // line by line, to make it hurt
    $lines = split("\n", file_get_contents($file));

    // per file, we'll set it once
    $account = '';

    foreach ($lines as $line ) {

        // see https://secure.php.net/manual/en/
book.simplexml.php#105330
        $lxml = simplexml_load_string($line);
        $json = json_encode($lxml);
        $array = json_decode($json,TRUE);

        // no attributes means no data, skip
        if (!isset($array['@attributes']))
            continue;

        $data = $array['@attributes'];

        // every file has a initial <session line containing
the account information
        if (strstr($line, '<session')) {
            if (isset($data['from'])) {
                $account = $data['from'];
            }
            // and since this is not a message, skip
            continue;
        }

        // no text, no need to keep this
        if (!isset($data['text']) || empty($data['text']))
            continue;

        if (isset($data['from_display']))
            $from = $data['from_display'];
        elseif (isset($data['to_display']))
            $from = $data['to_display'];
```

```
        elseif (isset($data['from']))
            $from = $data['from'];

        $time = date("Y-m-d H:i:s", $data['time']);

        _insert($time, $data['medium'], $account, $buddy,
$from, $data['text']);
    }
}
```

Pidgin

A few years ago I didn't appreciate plain text as much as I should. This was the era when the web exploded and everything used HTML or XML. *(And nowadays it's JSON and YAML everywhere, so we're still not entirely back to the elegant plain text solution, but much, much closer than we were.)*

The bad news about this is that I have mixed plain text and HTML logs for my Pidgin era; some are duplicates, some are not. The good news is that https://regex101.com/ exists.

Each file looks like this:

```
/{medium}/{account}/{contact}/{datetime format + timezone}.
{txt,html}
```

So getting the account and buddy information is easy and reliable.

chatlogs_pidgin_todo.sh

```bash
#!/bin/bash

cd pidgin
r=$(pwd)
r=${r//\//\\\/}
find . -iname *.txt | sed "s/\.\//$r\//g" > pidgin_todo
find . -iname *.html | sed "s/\.\//$r\//g" >> pidgin_todo
find . -iname *.html | sed "s/\.\//$r\//g" >> pidgin_todo
```

chatlogs_pidgin.php

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'pidgin' . DIRECTORY_SEPARATOR .
'pidgin_todo' ));

foreach ($files as $file) {
```

```
    if (empty($file))
        continue;

    $fdetails = pathinfo($file);
    $fdetails['dirname'] = str_replace('/absolute/path/to/
pidgin/','',$fdetails['dirname']);
    $details = split("/", $fdetails['dirname']);
    $medium = $details[0];
    $account = $details[1];
    $buddy = $details[2];
    $ftime = strtotime($fdetails['filename']);

    $c = file_get_contents($file);

    $matches = array();

    if ($fdetails['extension'] == 'txt') {
        preg_match_all('/\(([0-9]{2}:[0-9]{2}:[0-9]{2})\)
(.*?): ([^(]*)/sm', $c, $matches);
    }
    else {
        preg_match_all('/<font color="#.*?">.*?\((.*?)\).*?
<b>(.*?)<\/b>(.*)/m', $c, $matches);
    }

    if (empty($matches[0])) {
        continue;
    }

    foreach ($matches[0] as $key => $line ) {

        $text = $matches[3][$key];
        $ctime = strtotime($matches[1][$key]);
        $from = $matches[2][$key];

        $time = date('Y-m-d',$ftime) . " " . date('H:i:s',
$ctime);

        _insert($time, $medium, $account, $buddy, $from,
$text);
```

```
    };

}
```

Skype

Skype uses binary logs. In theory, there is Skype Historian[^13] and SkyNinja[^14] that could get the logs from Skype, but I have neither a Windows 7+ machine nor the patience to deal with this.

A long time I ago I cleaned my Skype history - I had a good reason - so it wouldn't really get me anywhere anyway.

Empathy

Empathy uses XML, but unlike Trillian's, these are valid XML files. Time & meta information is stored in the attributes, the actual text in the data.

Files look something like this: `{messed up account, protocol and medium name}/{buddy name}/{date}.log`

I decided to use a switch-case to tell the account and the medium, otherwise it's kind of impossible.

`chatlogs_empathy_todo.sh`

```bash
#!/bin/bash

cd empathy
r=$(pwd)
r=${r//\//\\\/}
find . -iname *xml | sed "s/\.\//$r\//g" > xml_todo
```

`chatlogs_empathy.php`

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'empathy' . DIRECTORY_SEPARATOR .
'xml_todo' ));

foreach ($files as $file) {

    if (empty($file))
        continue;

    $fdetails = pathinfo($file);
    $fdetails['dirname'] = str_replace('/absolute/path/to/
empathy/','',$fdetails['dirname']);
    $details = split("/", $fdetails['dirname']);
```

```
    $account = $medium = '';
    switch ($details[0]) {
        case 'butterfly_msn_user_40hotmail_2ecom1':
            $medium = 'msn';
            $account = 'user@hotmail.com';
            break;
        case 'gabble_jabber_xyz0':
            $medium = 'jabber';
            $account = 'xyz@gmail.com';
            break;
        case
'gabble_jabber_user_40domain_2ecom_40chat_2efacebook_2ecom0':
            $medium = 'facebook';
            $account = 'user@domain.com';
            break;
        case 'haze_icq__311111111':
            $medium = 'icq';
            $account = '11111111';
            break;
    }
    $buddy = $details[1];

    if (empty($medium) || empty($account))
        continue;

    $xml = simplexml_load_string(file_get_contents($file));

    foreach($xml->children() as $child) {
            $attr = $child->attributes();
            $attr = json_encode($attr);
            $attr = json_decode($attr,TRUE);
            $attr = $attr['@attributes'];

            $time = date("Y-m-d H:i:s",
strtotime($attr['time']));
            $from = $attr['name'];

            $text = $child->__toString();

            _insert($time, $medium, $account, $buddy, $from,
$text);
```

```
        }
}
```

Thunderbird

Thunderbird *(yes, it has built-in chat clients)* stores logs in line-by-line JSON. Broken, just like Trillian.

Files look something like this:

```
{medium}/{account}/{contact}/{full date with timezone}.json
```

Therefore getting medium, account and contact information is doable from the filename.

chatlogs_thunderbird_todo.sh

```bash
#!/bin/bash

cd thunderbird
r=$(pwd)
r=${r//\//\\\/}
find . -iname *.json | sed "s/\.\//$r\//g" > json_todo
```

chatlogs_thunderbird.php

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'thunderbird' . DIRECTORY_SEPARATOR .
'json_todo' ));

foreach ($files as $file) {

    if (empty($file))
        continue;

    $fdetails = pathinfo($file);
    $fdetails['dirname'] = str_replace('/absolute/path/to/
thunderbird/','',$fdetails['dirname']);
    $details = split("/", $fdetails['dirname']);
```

```
    $medium = $details[0];
    $account = $details[1];
    $buddy = $details[2];

    $lines = split("\n", file_get_contents($file));

    foreach ($lines as $line) {
        $data = json_decode($line);
        if (!is_object($data))
            continue;

        $time = date("Y-m-d H:i:s", strtotime($data->date));

        if (!isset($data->text))
            continue;

        if (isset($data->alias))
            $from = $data->alias;
        elseif (isset($data->title))
            $from = $data->title;
        else
            continue;

        $text = $data->text;

        _insert($time, $medium, $account, $buddy, $from,
$text);
    }
}
```

Hexchat and Xchat

Oh god.

This is when the real horror started. First of all Hexchat and Xchat has serious issues with their logging: logs go to `{account}/{contact,channel}.log`. No break per date at all, breaks are inside the files as `**** BEGIN LOGGING AT Thu Jul 30 09:15:56 2015`. There are differences in the display of from user, so it can be surrounded as:

- <user>
- .:user:.
- (user)

The dates are either `Sep 28 13:40:12` without year or full RCF `2015-10-26T12:49:35+0000`.

I also have 'sync-conflict' files in here: this is due to not waiting for full syncronization of syncthing before loading a client on another computer. This means I have a shitload of duplicates, but some of these sync conflict lines might contain useful information I don't have anywhere else.

I've tried automating some of the pre-cleanup tasks, but I've given this up. I ended up merging and renaming files by hand. The end goal was to have a structure of `{account}/{contact}.log`, where the per contact log contains everything I've found and merged; duplicates were expected and are taken care of anyway.

Unfortunately the log files are full of meta information as well I'm not interested in; those lines need to be skipped.

After the manual merges:

`chatlogs_hexchat_todo.sh`

```bash
#!/bin/bash

r=$(pwd)
r=${r//\//\\\/}
find . -iname *.log | sed "s/\.\//$r\//g" > ../hexchat_todo
```

`chatlogs_hexchat.php`

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

$files = split("\n", file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'hexchat' . DIRECTORY_SEPARATOR .
'hexchat_todo' ));

$medium = 'irc';

foreach ($files as $file) {
    // {account}/{contact}.log

    if (empty($file))
        continue;

    $fdetails = pathinfo($file);
    $account = end(split("/", $fdetails['dirname']));
    $buddy = $fdetails['filename'];

    $lines = split("\n", file_get_contents($file));
    $year = '2014';

    foreach ($lines as $line ) {
        if (empty($line))
            continue;

        $text = $from = $time = '';
        $message = $breaks = array();

        if (strstr($line, '▬▬▶'))
            continue;

        if (strstr($line, '◀▬▬'))
            continue;

        if (strstr($line, '▬▬▶'))
            continue;

        if (strstr($line, '◀▬▬'))
```

```
            continue;

        if (strstr($line, '<---    '))
            continue;

        if (strstr($line, '--->    '))
            continue;

        if (strstr($line, '(?)'))
            continue;

        if (strstr($line, '(☎)'))
            continue;

        if (strstr($line, '(✽)'))
            continue;

        if (strstr($line, '] is away') && strstr($line, '●'))
            continue;

        preg_match_all('/^.*? BEGIN LOGGING AT (.*)$/', $line,
$breaks);
        if (!empty($breaks[0])) {
            $year = date('Y', strtotime($breaks[1][0]));
        }

        // messages can be as:
        // - Sep 18 11:17:36 <user>    message
        // - Sep 21 21:56:15 .:user:.    message
        preg_match_all('/^([a-zA-Z]+) ([0-9]{2}) ([0-9]{2}:
[0-9]{2}:[0-9]{2})([^<>])[<.(]:?(.*?):?[>.)][ \t]?(.*)$/m',
$line, $message);

        if (!empty($message[0])) {

            $month = $message[1][0];
            $day = $message[2][0];
            $tstamp = $message[3][0];
            //10/Oct/2000:13:55:36 -0700
            $ctime = "{$day}/{$month}/{$year}:{$tstamp} +0000";
            $time = date('Y-m-d H:i:s', strtotime($ctime));
```

```
            $from = $message[5][0];
            $text = $message[6][0];
        }

        // messages can be as:
        // - 2015-10-26T12:49:35+0000 (user)    message
        preg_match_all('/^([0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:
[0-9]{2}:[0-9]{2}\+[0-9]{4})(.*?)[<.(]:?(.*?):?[>.)][ \t]?(.*)
$/m', $line, $message);

        if (!empty($message[0])) {
            $time = date('Y-m-d H:i:s', strtotime($message[1]
[0]));

            $from = $message[2][0];
            $text = $message[3][0];
        }

        if (!empty($from) && !empty($text) && !empty($time)) {
            _insert($time, $medium, $account, $buddy, $from,
$text);
        }
    }
}
```

Facebook

Sigh. I was foolish enough to sometimes use the online Facebook interface. There's only one way to get data out of there: [using their download].

That download gives you all you messages. In. A. Single. HTML. ALL. OF. THEM. I was starting to get furious, but instead of jumping into parsing a hideously large amount of data, I've found this: node.js convert-facebook-messages to JSON[^15]. While I don't like node I like when I don't need to write my own parser - and it works beautifully, generating a single JSON file.

Once this was done I had to handcraft two mappings: `Facebook ID => lowercase, noaccent, log-dir name long, nice, displayed name => lowercase, noaccent, log-dir name`

The first was relatively easy: since I have bitlbee running and doing Facebook, I just run:

`chatlogs_facebook_todo.sh`

```bash
#!/bin/bash

cat {zncdir}/moddata/log/{account}/bitlbee/status/
2015-12-23.log | awk '/\*\*\*\ Joins:/ { print $4,$5 } ' |
sort | uniq > /tmp/fbpeople
sed -i s/\(//g /tmp/fbpeople
sed -i s/\)//g /tmp/fbpeople
sed -i s/@facebook//g /tmp/fbpeople
cat /tmp/fbpeople | grep -v ^_ > /tmp/fbpeople2
```

and the required mapping is there, just needs formatting.

The good part of doing this is I could get rid of any previous occurrence of Facebook messages (where the medium was set to facebook, so I definitely have all the messages.

The second mapping I handcrafted: I got the output of the buddies from the script below, looked for the ones containing '@', copy-paste the ID to Facebook and see who it was. I could have automated this, but I believe that would have been much harder, as in my case, it was less than 10 people.

`chatlogs_facebook.php`

```php
<?php

function name_to_filename ($user) {
    $user = trim($user);
    $x = array (
        'Long Accented Name' => 'directorysafelowercasename',
    );

    if (isset($x[$user]))
        return $x[$user];

    return $user;
}

function id_to_name ($user) {

    $user = trim($user);
    $x = array (
        '100001111111111' =>  'directorysafelowercasename2',
    );

    $uid = str_replace('@facebook.com', '', $user);
    if (isset($x[$uid]))
        return $x[$uid];

    return $user;
}

function unset_me ($array) {

    $mynames = array (
        'Péter Molnár',
        'Molnár Péter',
        '[my fb uid]@facebook.com',
    );

    foreach ($mynames as $myname) {
        foreach ($array as $id => $var ) {
            $var = trim($var);
            if ($var == $myname) {
                unset($array[$id]);
```

```
            }
        }
    }

    return $array;
}

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );

// facebook.json is the one the converter did for me
$json = json_decode(file_get_contents(__DIR__ .
DIRECTORY_SEPARATOR . 'facebook.json'));

$medium = 'facebook';
$account = '[my facebook account]';

$channelcntr = 0;
$is_channel = false;

foreach ($json as $thread) {

    $users = split( ",", $thread->users );

    if (count($users) > 2 ) {
        $is_channel = true;
        $channelcntr = $channelcntr + 1;
        $buddy = "#facebook_group_chat-{$channelcntr}";
    }
    else {
        $users = unset_me($users);
        $is_channel = false;
        $buddy =
id_to_name(name_to_filename(trim(array_pop($users))));
    }

    $messages = $thread->messages;

    $prawtime = '';
    $psec = 0;
```

```
    foreach ($messages as $message) {
        $from = $message->user;

        // "Sunday, July 11, 2010 at 8:03am UTC+01"
        preg_match_all('/^([a-zA-Z]+), ([a-zA-Z]+) ([0-9]+),
([0-9]+) at (.*?) (.*?)$/', $message->date, $d );

        $year = $d[4][0];
        $time = date('H:i',strtotime($d[5][0]));
        $month = date('m',strtotime($d[2][0]));
        $day = $d[3][0];

        // magic to add seconds where there is no seconds
information
        $s = '00';
        $rawtime = "{$year}-{$month}-{$day} {$time}";

        if ($rawtime != $prawtime) {
            $sec = 1;
        }
        else {
            $sec = $psec + 1;
        }
        $prawtime = $rawtime;
        $psec = $sec;

        $sec = str_pad($sec, 2, '0', STR_PAD_LEFT);
        $time = $rawtime . ':' . $sec;

        $text = $message->text;
        _insert($time, $medium, $account, $buddy, $from,
$text);
    }
}
```

Generating output

chatlogs_output.php

```php
<?php

require_once ( __DIR__ . DIRECTORY_SEPARATOR .
'proc_tools.php' );
global $mysqli;

$processed = __DIR__ . DIRECTORY_SEPARATOR . '_processed';

if (!is_dir($processed))
    mkdir ($processed);

$res = $mysqli->query("SELECT * FROM `chatlogs` ORDER BY time
ASC");
while ($row = $res->fetch_assoc()) {
    echo "doing {$row['id']}\n";

    $adir = $processed . DIRECTORY_SEPARATOR . $row['account'];
    if (!is_dir( $adir ))
        mkdir ($adir);

    $bdir = $adir . DIRECTORY_SEPARATOR . $row['buddy'];
    if (!is_dir( $bdir ))
        mkdir ($bdir);

    $d = split(" ", $row['time']);
    $date = $d[0];
    $time = $d[1];

    $f = $bdir . DIRECTORY_SEPARATOR . $date . '.log';

    $fp = fopen($f,'a');

    $l = "[{$time}] <{$row['from']}> {$row['text']}\n";

    fwrite($fp, $l);
```

```
    fflush($fp);
    fclose($fp);
    touch($f, strtotime($row['time']));
    touch($bdir, strtotime($row['time']));
}
```

Remarks

Apparently both plain text and structured data can be done good and bad. Using XML or JSON while breaking these formats up to line by line parsing is bad, but not splitting text files at any given time is just as bad.

What's always bad is binary logging. Even if you have an utmost reason for binary logs - like there is no other format supported - come up with a way to export it either on the fly or at last when you are about to leave the software, because later than that it's going to be really painful. Any text format is better than unreadable, compressed, potentially encrypted binary.

I also observed the fact that some buddies needs merging, since the are the same people, just different accounts. I'd do this on the database or later, manually, on the generated texts.

As a result, after getting rid of publicly logged rooms (an additional ~300k lines), I ended up with ~200k usable lines; exported to the common format it's ~20MB of plain text.

Links

1. http://john-114.narod.ru/fido/7.jpg
2. http://wiki.znc.in/ZNC
3. http://bitlbee.org/
4. https://syncthing.net/
5. http://www.msgplus.net/
6. https://www.trillian.im/
7. https://pidgin.im/
8. http://skype.com/
9. https://wiki.gnome.org/Apps/Empathy
10. https://www.mozilla.org/en-US/thunderbird/
11. https://github.com/hexchat/hexchat
12. https://facebook.com
    ``
13. http://gh.eigenein.xyz/skype-historian/
14. http://gh.eigenein.xyz/skyninja/
15. https://github.com/hey-johnnypark/convert-facebook-messages