検索機能を付けた個人別履歴の作成

96H068 高徳　哲也

目 次

1　はじめに

ここ数年のインターネットの普及により大学内などでも Web ページを閲覧することによって情報を取得する機会が増えてきており、それに伴い Web ページを閲覧する回数も多くなってきている。そこでユーザが過去に見た Web ページを再び見るためによく使われる機能として WWW ブラウザのブックマーク機能があるが、このブックマーク機能には欠点もある。面倒なブックマーク登録作業やブックマークの整理などがあり、特に問題となるのがブックマークを利用する際使用できるパソコンが限定されることである。この事は大学などのパソコンが複数台ある演習室などでは大きな問題となる。

そこで本研究では個人別の履歴表示と検索機能を用いることによってユーザが大学などのパソコンが複数台あるような演習室などで使用するパソコンを限定されず、より簡単に過去に見た Web ページを見つけ出せる環境を提供することを目的とした。

1.1　従来のブックマーク機能

従来のブックマーク機能について

- 利点

  ・自分用に最適化されたディレクトリ型サーチエンジンの作成

  ブックマークのメニューはブックマークに登録されたタイトルをディレクトリ型に保存していくことができるようになっている。このためブックマークのメニューをユーザが使用しやすいようにカスタマイズしていくことができる。

  ・Web ページを移動する際 URL を入力せずに済む

  ブックマークに登録するとユーザは URL を入力せずに目的の Web ページに移動することができる。これはブックマーク機能の大きな利点で、これによってユーザは目的の Web ページを見るために URL をわざわざ入力しなくてもブックマークに登録されたタイトルを選ぶだけで見たい Web ページを閲覧することができるようになる。

- 欠点

  ・面倒なブックマーク登録作業

  ブックマークに登録するには Web ページごとにブックマークに追加するという作業をしなければならない。そのため Web ページを見ている時点ではそのページが大事ではないと思いブックマークに登録せずに、後日になって再びその Web ページを見たいと思った時はブックマークに登録していないため goo などの検索エンジンを使うなどして Web ページを探さなければならず多くの時間と労力を費してしまう。

  ・面倒なブックマークの整理

  ブックマークは登録するたびにタイトルが増えていき見づらくなっていく。そのため、定期的に整理せねばならず面倒な作業をしなくてはならなくなる。

  ・ブックマークを利用する際使用できるパソコンが限定される

  特に問題となるのがこのブックマークを利用する際使用できるパソコンが限定されることである。通常ユーザがブックマークを登録すると次からブックマークを利用するためにブックマークを登録したパソコンを利用しなければならない。決められたパソコン一台を一人で使う場合には問題ないが、パソコンが複数台あるような演習室ではユーザ一人一人に決められたパソコンがあるわけではないので毎回同じパソコンを使用することはない。そのためブックマークを登録し利用することが困難となってしまう。

1.2 本研究で作成した検索機能を付けた個人別履歴の特徴

- 本研究での利点

  本研究での大きな利点はブックマーク機能の欠点である大学などの演習室で使える点である。これによってユーザは演習室などでどのマシンを使っても自分自身の履歴を見ることが可能となる。また、もう一つの利点はブックマーク機能の欠点である面倒な登録作業やブックマークの整理をやらなくてもよいことである。これによってユーザは面倒な作業をせずに見たい Web ページを閲覧することができる。履歴表示にはタイトルだけでなく内容や Web ページを見た時間も表示させることによってよりわかりやすくユーザに見たい Web ページを閲覧できる環境を提供できるようになっている。

- 本研究での欠点

  本研究では個人別履歴にキャッシュサーバの情報を元に作成しているため、前提としてキャッシュサーバもしくはプロクシサーバが設置していなければならない。また個人別履歴とキーワード検索には CGI を用いて行うため、CGI サーバが設置していなければならない。またもう一つの欠点として保存しておく履歴の長さがある。ブックマークの場合保存するデータの量はそれほど多くないが本研究の個人別履歴は見た Web ページの全てが保存されるのでデータの量が多くなってしまう。そのため保存できる履歴の長さが限られてしまう。本研究では一週間分の履歴しか見ることができない。

2　個人別履歴

本研究では従来のブックマークの欠点を補うためにキャッシュサーバに保存してある様々なログを用いて、フィルタ処理を行い個人別の履歴を作成する。履歴表示には web ページを見た時間、タイトル、内容の表示をさせてユーザが探している Web ページをより探しやすく表示させる。そのため、キャッシュサーバに保存してあるアクセスログなどのログから履歴に必要なユーザ情報、Web ページにアクセスした時間、URL、Web ページのタイトルや内容を取り出すのにフィルタ処理を行ってその日の全ユーザの履歴を作り、ユーザ個人の履歴表示にはその日の全ユーザの履歴を元に CGI を用いてユーザからの要求があるたびに作成する。

2.1　ログのフィルタ処理

本研究ではキャッシュサーバとして使われている Squid のログを用いて個人別の履歴を作成した。Squid は、FTP や HTTP データオブジェクトに対応した、Web クライアントのための、高性能な代理キャッシュサーバである。この Squid には全てのアクセスを記録することもサポートしており、履歴作成に必要な情報が保存されている。この情報にフィルタ処理を行ない個人別の履歴を作成する。

2.1.1　Web ページのタイトル、内容の取得

store.log・cache にある情報をフィルタ処理にかけて Web ページのタイトル、内容を取り出す。この時 cache には HTML で書かれた情報以外にも画像情報なども保存されているので store.log に保存されている情報から HTML で書かれた情報のみを取り出すようにする。

1. store.log

   store.log は以下のような情報が保存されている。以下では 2 行にわたって書かれているが本来は 1 行で書かれている。

   ```
   949604672.484 SWAPOUT 00000F57  200 -1 949901686 -1 text/html 6183/6183 GET
   http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000_maidome086.html
   ```

   以上の情報から履歴作成に必要な情報をフィルタ処理して取り出す。

   - 00000F57 Web ページの情報がどの場所に保存されているかを示す。
   - text/html この情報をもとに HTML で書かれた情報のみを取り出す。
   - http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000 _maidome086.html Web ページの URL

2. cache

   cache は以下のような情報が保存されている。

   ```
   YFF><FF><FF>vY<9E>8^@^@^@^@^A^@‘^D^DJ^@^@^@http://news.yahoo.co.jp/headlines/mai
   <9E>8^@^@^@^@^A^@‘^D^DJ^@^@^@http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000_maidom
   e086.html^@^HHTTP/1.0 200 OK
   Last-Modified: Mon, 07 Feb 2000 05:34:46 GMT
   Content-Type: text/html
   Content-Length: 6183
   Age: 0
   X-Cache: MISS from EmpokNor.osaka-sandai.ac.jp
   ```

```
Proxy-Connection: keep-alive

<html>
<head>
<!-- <FD><FE> -->

<title>Yahoo! - ＜フグ中毒＞釣ったフグ食べた６人が中毒　東京都江戸川区で＝替（毎日新聞）</tit
le>
</head>
<body>
<center>
<table cellpadding=2 cellspacing=0 width="100%">
  <tr>
  <td rowspan=2 width="1%"><a href="/headlines/">
  <img width=248 height=40 border=0 src="http://news.yahoo.co.jp/images/jp_news4.gif"  alt="Ya
hoo!ニュース">

</center><hr><b>2000 年 2 月 7 日（月） 14 時 27 分</b>
<h2>＜フグ中毒＞釣ったフグ食べた６人が中毒　東京都江戸川区で＝替（毎日新聞）</h2><br>
<!--j_mai|949901220-->
<!-- TextStart -->

<br>
  ６日午後８時１５分ごろ、東京都江戸川区内の病院から「フグ中毒の患者が６人いる」と、転送先の
病院を照会する１１９番通報があった。
```

以上の情報から履歴作成に必要な情報をフィルタ処理して取り出す。<title> から </title> までに書かれた部分からはタイトルを、<body> から以下タグを抜き取った部分からは内容を取り出す。

store.log cache にフィルタ処理を行った結果

```
 http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000_maidome086.html  |
Yahoo! - ＜フグ中毒＞釣ったフグ食べた６人が中毒　東京都江戸川区で＝替（毎日新聞）|
釣ったフグ食べた６人が中毒　東京都江戸川区で＝替（毎日新聞）６日午後８時１５分ごろ、東京都江戸川区内
```

上記の結果を process1 というファイル名で保存する。

2.1.2 一日分の全履歴を作成

access.log は以下のような情報が保存されている。

```
949608164.132  32779 192.168.1.91 TCP_MISS/200 6432 GET
http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000_maidome086.html  takatoku
FIRST_PARENT_MISS/proxy-omp.osaka-sandai.ac.jp  text/html
```

以上の情報から履歴作成に必要な情報をフィルタ処理して取り出す。

- 949608164 Web ページを見にいった時間
- http://news.yahoo.co.jp/headlines/mai/000207/dom/14270000 _maidome086.html Web ページの URL
- takatoku Web ページを見たユーザ

prosess1 に access.log にフィルタ処理を行った結果を組み合わせる。

```
takatoku | 192.168.1.151 | 18:20:35 | http://news.yahoo.co.jp/headlines/hwj/000128/cpt/15413500_wircpt007.html  |
（Yahoo! - 米陸軍と NASA がシミュレーション技術を共同で開発（WIRED）|
トップ Yahoo!JAPAN- カスタマイズ-ヘルプトピックストップ国内海外経済企業マーケットエンターテインメント
```

上記の結果を process2 というファイル名で保存する。

2.2 個人別の履歴を作成

ユーザ認証とログ内に保存されているユーザ情報の作成は WWW ブラウザの環境変数を用いて行っている。そのため本研究ではこの環境変数に用いられる情報がクライアントマシン側にあることが前提となるが、これを用いることによってユーザはユーザ認証のためにユーザ名やパスワードを入力すること無く個人の履歴を作成し、見ることができる。

2.2.1 ユーザ識別の方法

本研究で用いたユーザ識別には WWW ブラウザの環境変数である REMOTE_IDENT を利用している。REMOTE_IDENT とは WWW サーバで IDENT が実行されていた場合、クライアントマシンの IDENT_DAEMON によって返される値のことである。この REMOTE_IDENT をユーザ識別に用いることによってユーザはユーザ識別のためにユーザ名やパスワードを入力することなく個人の履歴を見ることができる。ただしこの環境変数を利用するには以下の作業をしなければならない。

Squid の設定で /etc/squid/squid.conf を

#ident_lookup_access deny all から

ident_lookup_access allow all に変更する。
HTTP サーバの設定で /etc/httpd/conf/http.conf を以下のようにする。

```
#  TAG: ident_lookup_access
#       A list of ACL elements which, if matched, cause an ident
#       (RFC 931) lookup to be performed for this request.  For
#       example, you might choose to always perform ident lookups
#       for your main multi-user Unix boxes, but not for your Macs
#       and PCs.  By default, ident lookups are not performed for
#       any requests.
#
#       To enable ident lookups for specific client addresses, you
#       can follow this example:
#
#       acl ident_aware_hosts src 198.168.1.0/255.255.255.0
#       ident_lookup_access allow ident_aware_hosts
#       ident_lookup_access deny all
#
#       This option may be disabled by using --disable-ident with
#       the configure script.
#ident_lookup_access deny all
```

次の行に、ident_lookup_access allow all を付け加える。

以下にサーバマシンがクライアントマシンからユーザ名を取得する仕組みを図を使って説明する。

図 1: IDENT_DAEMON

1. クライアントからサーバに対して Web ページの要求をする。
2. サーバはクライアントの IDENT_DAEMON 対してユーザ名の問い合わせをする。
3. クライアントはサーバに対してユーザ名を返す。これによってサーバ側の環境変数の一つである REMOTE_IDENT に接続ユーザ名が入る。

2.2.2　個人別履歴の表示結果

個人別履歴の表示結果にはユーザ名と履歴が表示される。履歴には Web ページを見た時間、URL、タイトル、内容が表示される。

図 2: 個人別履の表示結果

3　キーワード検索機能

また履歴表示に Web ページに含まれていたキーワードを用いて検索できる機能を付け加えたものを作成する。このキーワードを用いた検索機能には日本語全文検索ソフトである「Namazu」を用いた。本研究での検索機能にはキャッシュサーバのログに保存している URL のみを検索し、検索する日付を指定することにより、より的確に捜し出すことができ、検索結果から目的の Web ページが簡単に閲覧できるようになっている。

3.1　Namazu について

Namazu とは、手軽に使えることを第一に目指した日本語全文検索システムである。この Namazu を CGI として動作させることにより、小中規模の WWW 全文検索システムを構築できるほか、ハードディスク内のファイルを対象としたようなパーソナルな用途にも使えるようになっている。またこの Namazu はフリーソフトウェアである。

3.1.1　Namazu の特徴

Namazu の特徴

1. 各種プラットホームに対応

   Namazu は、Linux、FreeBSD、Solaris、SunOS、IRIX、HP-UX、OSF/1 などの UNIX 系プラットホーム全般で動作させることができる。また、Window98 や WindowsNT など Windows プラットホーム上でも動作するバージョンが移植されており、バイナリ配布もされている。これによって Namazu を動作させる時、プラットホームを限定されずに動作させることが可能となる。ただし、現時点では MS-DOS、Macintosh プラットホームでは動作しない。なお本研究では Namazu を Linux 上で動作させた。

2. 高速な検索を実現

   ファイルごとに文字列一致を行う grep などの原始的な検索方法とは異なり、Index の作成を行なう全文検索システムの構造をとることによって圧倒的に速く検索結果を得ることができるのがこの Namazu の最大の特徴となっている。例えば目安として、Pentium120MHz、メモリ 64MB の Linux マシンで 50MB のファイルを基に作成されたインデックスを検索した場合、だいたい 0.1 秒程度で検索が完了する。検索が終了して結果を表示するまでを含めても 1 秒もかからない。

3. 多くの検索方法をサポート

   検索方法としては、ＡＮＤ・ＯＲ・ＮＯＴ検索などデータベース検索における基本的なものをサポートしている。また、フレーズ検索および中間一致、後方一致、正規表現による検索もサポートしているなど、ユーザのニーズに応じた自由度の高い検索を簡単に行うことができるようになっている。

   本研究では検索方法を詳しく書いたものをキーワード検索のトップページで説明している。

4. 検索結果表示の柔軟性

   検索結果表示は goo や infoseek などの一般のインターネットサーチエンジンと同じ感覚で操作できる。例えば、通常は結果表示が URL と同時に要約も表示されるが、要約が不要な場合にはその有無を簡単に切り替えることが出来る。また、検索結果が多くヒットした場合などはページ単位で表示されるようになっている。

   本研究では標準で要約表示が表示され、２０件単位で検索結果が表示されるようになっている。

5. 正確な HTML の取り扱い

   インデックス作成の際には、HTML ファイルに含まれているタグから ALT 要素を取り出したり、検索結果の表示に JIS 漢字コードで HTML を出力するなど、HTML の扱いにはできるだけ配慮している。

   本研究ではキーワードの検索対象が HTML で書かれたファイルのため、この特徴が一つの大きな利点となっている。

6. 複数インデックスからの検索をサポート

   Namazu では、あらかじめ作成しておいた複数のインデックスに対して同時に検索を行い、結果をマージして表示することができる。このため、あらかじめおおまかな分野別のインデックスを作成しておけば、きめ細かな検索をすることが可能になる。

   本研究ではインデックスを一日ごとの履歴で作っているため複数のインデックスを作成している。

7. 多彩な詮索クライアント

   Namazu の最も単純な検索の形態は、コマンドラインでキーワードを与えて行なうものである。例えば UNIX プラットホームの場合は、kterm などのターミナルエミュレータを開いてコマンドとキーワードを入力して実行する。これは Namazu のもともとの目的が、Web サイト上で CGI を設定して検索するものであったことに由来する。また、GUI スクリプト言語である Tcl/Tk を利用した検索クライアントである「TkNamazu」を使って X Window System でも利用できるようになっている。これらは全て内部的にコマンドラインからの検索コマンドを呼び出して、使いやすいユーザインタフェースを提供している。このように Namazu のサポートする検索方法が CGI による http 経由の検索ばかりではないことは、他のフリーの全文検索にはない大きな特徴となっている。

   本研究では演習室などで誰でも簡単にキーワード検索ができることを目的としているため、Web サイト上で GUI 操作ができ操作が簡単な CGI を検索クライアントとして用いている。

3.1.2 Index の作成

Index 作成は実際の検索作業を始める前にあらかじめ Index を作成しておく作業である。この Index とは書籍の索引を意味する。全文検索システムの Index も、書籍の索引とまさに同じ機能を持っている。全文検索システムの場合、Index 作成で作られる Index はどの単語がどの文書 (Web ページ) に含まれているかを示すものである。通常の本の場合には、重要と思われる単語についてのみ著者がいくつか抜き出し、その単語についての Index だけが作成される。しかし、全文検索システムの場合は文書に含まれる全ての単語についての Index 作成を行う。全ての文書に Index を作成することは一見無駄に思えるが、検索時の処理はコンピュータが行うため Index の容量を若干増やす事になるが、そのことが検索スピードを遅くするという事にはならない。

Index 作成コマンドである mknmz を実行すると、ディレクトリに NMZ.*というファイルが数多く作成される。これらのファイル群は全て Index としての役割を果たすので、一つでも欠けてしまうとうまく動作しなくなってしまう。特に重要なファイルは以下で説明するファイルのうち上から 6 つまでのファイルであり、これらのファイルは削除しても変更してもいけない。その他のファイルは検索結果画面表示のカスタマイズに用いたり、検索されたキーワードのログを記録するファイルだったり、namazu が内部的に用いるファイルである。

表 1 :インデックスファイル群 (NMZ.*) の説明

ファイル名	内容
NMZ.i	インデックスファイル (転置ファイル、inverted ファイル)
NMZ.ii	インデックスファイル seek 用インデックス
NMZ. ｆ	ファイルリスト・ファイル (各ファイルの情報を記録)
NMZ.fi	ファイルリスト・ファイル seek 用インデックス
NMZ.h	キーワードの先頭 2 バイト用のハッシュテーブル
NMZ.r	インデックスに登録されているファイルのリスト
NMZ.head.en	検索結果出力用ヘッダファイル (英語)
NMZ.head.ja	検索結果出力用ヘッダファイル (日本語)
NMZ.foot.en	検索結果出力用フッタファイル (英語)
NMZ.foot.ja	検索結果出力用フッタファイル (日本語)
NMZ.msg.en	ロック時のメッセージ用ファイル (英語)
NMZ.msg.ja	ロック時のメッセージ用ファイル (日本語)
NMZ.body.en	キーワードが与えられなかったときのメッセージ (英語)
NMZ.body.ja	キーワードが与えられなかったときのメッセージ (日本語)
NMZ.log	インデックスの更新ログ
NMZ.slog	検索されたキーワードのログ
NMZ.lock	ロック用ファイル
NMZ.le	little-endian なインデックスのときに存在
NMZ.be	big-endian なインデックスのときに存在
NMZ.w	正規表現・中間/後方一致用の単語表
NMZ.p	フレーズ検索用のインデックス
NMZ.pi	フレーズ検索用のインデックスのインデックス

3.2 Namazu のカスタマイズ

本研究でキーワード検索をより快適に行うために Namazu をカスタマイズした。

まずは検索する文書群 (Web ページ) のインデックスを作成しなくてはならない。検索対象である Web ページは cach に保存されているが全てにインデックスを作成すると画像データなど無駄な情報までインデックスを作成してしまうため store.log にフィルタ処理を行い HTML で書かれた文書のみにインデックス作成を行う。

またより的確に見たい Web ページを捜し出すために検索する日付を指定するため日付ごとにインデックスを作成するため複数インデックスを作成するようにしなくてはならい。そのためキーワード検索のメニュー画面も分かりやすいように作成した。

3.3 キーワード検索

3.3.1 メニュー画面

キーワード検索のメニュー画面には検索式を入力するメニューのほかに、表示件数・要約表示・対象インデックスを選ぶメニューがある。

検索式の入力方法に関してはメニュー画面の中で説明している。表示件数は 10,20,30,50,100 と選択でき、内容の一部を表示する要約表示は ON,OFF が選択でき、対象インデックスは当日を入れて 7 日間の中から選択できるようになっている。

図 3: キーワード検索によるメニュー画面

3.3.2　検索結果による画面

検索結果には参考ヒット数と検索式にマッチした文書が表示される。

参考ヒット数は検索の対象となった日付にそれぞれどれほど文書がマッチしたかを表示したものである。

表示結果の順序は、キーワードを含む文書に対してそれぞれの文書がどの程度検索条件にマッチしてるかを示す指標として「スコア」と呼ばれる点数付けを行い、そのスコア順に並べ替えたものを検索結果として出力する。これを「スコアリング」、または「ランキング」という。一般的には

- 検索語の頻度
- ファイルサイズ
- 文書構造

などを利用して総合的に評価してスコアリングを行う。つまり、キーワードとなる文字列が複数回出現している文書あるいは、キーワードとなる文字列をタイトルや見出しの中に含んでいる文書には高い点数が与えられ、上位に位置する。タイトルの横に書かれている score:の数字がそれぞれのスコアである。

図 4: キーワード検索による表示結果

4 考察

本研究で作成した個人別の履歴表示と検索機能を用いることによってユーザは演習室などで使用するパソコンを限定されず、より簡単に過去に見た Web ページを見つけ出せることが可能になった。

しかしながら、いくつかのプログラム上の問題点などが残った。

- Web ページによってはキャッシュに含まれているバイナリデータによる履歴のタイトル、内容表示での文字化け問題

  これはフィルタ処理のプログラムの問題で、全ての cache にバイナリデータが含まれているわけではないので解決できなかった。ただしこれはプログラム上の問題であるためプログラムを書き換えれば解決すると思われる。

- 大量に履歴が増えると、Namazu によるインデックス作成に時間がかかる問題

  これは cache にある HTML の文書全てに Index を作成するためユーザの数や Web ページの閲覧数が増えると起こる問題で、その日の Index 作成はデフォルトで 30 分ごとにしてあるが作成に時間がかかるとその時間で終らない問題が起こる。これは対象となる Web ページを限定して例えば access.log でユーザ情報を持たないログも存在するためその情報を元に誰が見たかわからない Web ページは Index を作らないなどすれば解決すると思われる。

- クライアントマシンの OS によってはユーザ識別のために用いる情報を標準で持たないものもあり、その場合クライアントマシンごとに情報を持たせるよう設定を変えなければならない問題

  この問題は本研究で最も大きな問題で、演習室などのようにクライアントマシンが数多くある場合は一台一台に設定をせねばならず、またキャッシュサーバに関しても同様に設定を変えなければこの情報を受け取ることができず面倒な作業をしなければならない。

5 まとめ

本研究では、キャッシュサーバとして使われている Squid を利用してログにフィルタ処理を行い、CGI の環境変数を用いることによってユーザ別の履歴を表示させることができた。さらにキーワード検索機能を付けることによってより簡単に過去に見た Web ページを再び閲覧することが可能になった。

しかしながらいくつかのプログラム上の問題点などが残るなど解決しなければならない問題があるため本研究で作成したものは改善の余地がある。

さらに検索機能をより充実するために時間帯指定での URL 検索やユーザによる履歴の保存期間の指定することなどを実現することが今後の課題として考えられる。

謝辞

本研究を進めていく上で様々な御協力及び、御指導頂いた藤井 信教授、大垣 斉講師ならびに中村 孝講師そして各研究室の方々に深く感謝の意を表します。

参考文献

- 『プログラミング言語 AWK』 A.V. エイホ/B.W. カーニハン/P.J. ワインバーガー = 著, 足立 高徳 = 訳
- 『日本語全文詮索システムの構築と活用』 馬場 肇 = 著
- 『AWK を 256 倍使うための本』 志村 拓/鷲北 賢/西村 克信 = 著

付録

ログのフィルタ処理を行なうファイル log3

```
#!/bin/awk −f



{line[NR] =$3
 line2[NR] = $8
 line3[NR] = $11
}



END{
    for (i = 0; NR > i; i = i + 1){
        if(line2[i] ∼ /html$/)
        if(line[i] !∼ /ˆF/){
          a = substr(line[i],3,2)
          b = substr(line[i],5,2)

          #cache のパスを指定する。
          pass = ENVIRON["cache"]a"/"b"/"line[i]

#=================================================
#タイトルを取得する

          x = "tr -d \"\r|\n\" < "pass
          if((x | getline ) > 0){
            IGNORECASE = 1
            b=index($0,"<TITLE>")
            d=substr($0,b+7)
            u=index(d, "</TITLE>")
            u=u−1
            tmp=substr(d,1,u)

#=================================================
#内容を取得する

            a=index($0,"<BODY")
            s=substr($0,a)
            gsub("<[ˆ>]*>","",s)
            gsub(" ","",s)
```

```
          s=substr(s,1,100)
          if(b == 0 || u == 0)

#=====================================================
#結果を出力する

            printf("%s | (No title) | %s\n\n",line3[i],s) | "nkf -e"
          else if(a == 0 )
            printf("%s | (%s) | 内容なし\n\n",line3[i],tmp) | "nkf -e"
          else
            printf("%s | (%s) | %s\n\n",line3[i],tmp,s) | "nkf -e"

        }
        close("nkf -e")
        close(x)
      }
    }
}
```

一日分の全履歴を作成するファイル log4

```
#!/bin/awk −f

{
  line[NR]  = $1
  line2[NR] = $3
  line3[NR] = $7
  line4[NR] = $8
}

END{
  for(i = 0; NR > i; i = i + 1){

    #access.log から HTML で書かれたログのみを取り出す。
    if(line3[i] ~ /html$/ || line3[i] ~ /htm$/ || line3[i] ~ /\/$/){
      #process1 のパスを指定する。
      x = "grep " "'"line3[i]"' " ENVIRON["pro2"]"process1"
      if((x | getline) > 0){
```

```
#=================================================
#UNIX TIME を JST に変換して表示する。タイトル、内容がある場合。

        printf("%s | %-15s |%2d:%02d:%02d | %s\n",
      line4[i],
        line2[i],
        (line[i] + 32400) % 31536000 % 86400 / 3600,
        line[i] % 31536000 % 86400 % 3600 / 60,
        line[i] % 60,
        $0)

      }

#=================================================
#UNIX TIME を JST に変換して表示する。タイトル、内容がない場合。

      else printf("%s | %-15s | %2d:%02d:%02d | %s | \n",
      line4[i],
      line2[i],
      (line[i] + 32400) % 31536000 % 86400 / 3600,
      line[i] % 31536000 % 86400 % 3600 / 60,
      line[i] % 60,
      line3[i])
       close(ENVIRON["pro2"]"prosess1")
       close("grep ")
       close(x)
     }
  }
}
```

log3,log4 を実行するスクリプトファイル log13

```
#!/bin/sh

. /home/takatoku/bin/settei.conf

$log"log3" $store > $pro2"process1"
$log"log4" $access > $pro2"process2"
```

履歴を保存するファイル log14

```
#!/bin/sh

. /home/takatoku/bin/settei.conf

cd $pro1
mv 5/* 6/
mv 4/* 5/
mv 3/* 4/
mv 2/* 3/
mv 1/* 2/
cp -r today/* 1/
```

Index ファイルを保存するファイル log15

```
#!/bin/sh

. /home/takatoku/bin/settei.conf

cd $index1

mv 4/ 5/
mv 3/ 4/
mv 2/ 3/
mv 1/ 2/
mv today/ 1/

rm −r 5/
rm 4/NMZ.*
rm 3/NMZ.*
rm 2/NMZ.*
rm 1/NMZ.*

mknmz −O $index1"4/" $index1"4/"
mknmz −O $index1"3/" $index1"3/"
mknmz −O $index1"2/" $index1"2/"
mknmz −O $index1"1/" $index1"1/"
```

設定ファイル settei.conf

```
export cache="/var/spool/squid/"                #squid のキャッシュディレクトリがあるパス
export access="/var/log/squid/access.log"       #squid の access.log がある絶対パス
export store="/var/log/squid/store.log"         #squid の store.log がある絶対パス

export log="/usr/local/bin/"                    #実行ファイルがある絶対パス
export pro1="/home/takatoku/public_html/"       #process1・process2 ファイルが保存されているディレクトリがある絶対パス
export pro2="/home/takatoku/public_html/today/"  #当日の process1・process2 ファイルがある絶対パス
export index1="/home/takatoku/public_html/index/" #Index ファイルが保存されているディレクトリがある絶対パス
export index2="/home/takatoku/public_html/index/today/"  #当日の Index ファイルがある絶対パス
```

個人別の履歴を作成するファイル usr0.cgi

```
#!/bin/awk −f

BEGIN{
   print "Content-type: text/html\n\n"
   print "<html>\n<head>\n<title>個人別履歴</title>\n"
   print "<meta http-equiv=Content-Type content=text/html; charset=x-euc-jp>\n"
   print "</head>\n"
   print "<body bgcolor=#336699 text=#FFFFFF link=#FFFFFF vlink=#FFFFFF alink=#FF0000>\n"

   usr = ENVIRON["REMOTE_IDENT"]" "
   personal = "/home/takatoku/public_html/today/process2"
   FS = "|"

     printf "<font size=5>個人別履歴 ( %s)<SPACER SIZE= "50" TYPE="horizontal">  \n", usr
   print date | "date"
   print "</font><BR><HR SIZE=3><BR>"

   print "<A HREF=\"usr0.cgi \">[today]</A><A HREF=\"usr1.cgi \">[1]</A> <A HREF=\"usr2.cgi \">[2]</A>
<A HREF=\"usr3.cgi \">[3]</A> <A HREF=\"usr4.cgi \">[4]</A> <A HREF=\"usr5.cgi \">[5]</A> <A HREF=\"usr6.cgi
\">[6]</A> <BR>"
   print "</font><BR><HR SIZE=3><BR><BR>"
```

```
    while((getline < personal) > 0)
        {
                if($1 == usr)
                {
                    printf "%s %s<BR> <a href=\"%s\" TARGET=_blank>%s<\/a><BR>%s<BR><BR>\n",$3,$5,$4,$4,$6

        }

    }

}

END{
    print "\n</body>\n</html>"

}
```

キーワード検索によるメニュー画面 search.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META HTTP−EQUIV="Content-Type" CONTENT="text/html; charset=ISO-2022-JP">
<LINK REV=MADE HREF="mailto:webmaster@localhost">
<TITLE>Namazu search system for Vine Linux</TITLE>
</HEAD>

<body bgcolor="OldLace" LANG="ja">

<table width=640><tr>

<td width=490>
<font size=6 color="#003300">
<H1> キーワード検索 </H1>
</font>
</td>
```

```
</tr>
</table>

<hr>

<blockquote>
<h4> キーワードを入力してください </h4>
<FORM METHOD="GET" ACTION="http://www.fken/~takatoku/cgi-bin/namazu.cgi">
<P>
<STRONG> 検索式:</STRONG>
<INPUT TYPE="TEXT" NAME="key" SIZE="40">
<INPUT TYPE="SUBMIT" NAME="submit" VALUE="Search!">
<INPUT TYPE="HIDDEN" NAME="whence" VALUE="0">
</P>
<P>
<STRONG> 表示件数:</STRONG>
<SELECT NAME="max">
<OPTION VALUE="10">10
<OPTION SELECTED VALUE="20">20
<OPTION VALUE="30">30
<OPTION VALUE="50">50
<OPTION VALUE="100">100
</SELECT>
<STRONG> 要約表示:</STRONG>
<SELECT NAME="format">
<OPTION SELECTED VALUE="long">ON
<OPTION VALUE="short">OFF
</SELECT>
</P>
<INPUT TYPE="HIDDEN" NAME="whence" VALUE="0">


<STRONG> 検索対象 </STRONG><BR>
 <UL>
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="today" CHECKED> 今日
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="1" CHECKE>1 日前
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="2" CHECKE>2 日前
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="3" CHECKE>3 日前
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="4" CHECKE>4 日前
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="5" CHECKE>5 日前
  <LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="6" CHECKE>6 日前
```

</UL>

</FORM>

</blockquote>

<hr>

<H2> 検索の方法 </H2>

<P>
大文字、小文字の区別はありません。<VAR>foo*</VAR> のように末尾にアスタリスクを指定することで前方一致検索が可能です。また、単語をスペース区切りで並べて書くとアンド検索になります。日本語は KAKASI/ChaSen によって分解され、「日本語情報処理」なら「日本語」「情報処理」のように 2 つの単語に分かれてアンド検索されます。日本語の単語の分解は完全ではありません。品質は辞書によって決定されます。
</P>

<P>
全角 (2 bytes) アルファベット・記号はすべて 1 byte として処理されます。記号を含む検索も可能で <CODE>TCP/IP</CODE> というような単語の検索も可能です。ただし、記号の処理は完全ではないので <CODE>TCP IP</CODE> のように分けてアンド検索をかけた方が取りこぼしがありません (その代わり余計なファイルまでヒットしてしまう可能性もありますが)。
</P>

<P>
括弧を含めたアンド検索とオア検索およびノット検索が可能になっており検索式に <CODE>& | ! ( )</CODE> を用います。記号の代わりに <CODE>and/or/not</CODE> で指定することも可能です。
検索式はひとつづつスペース区切りで入力しなければなりません。
たとえば
</P>

<UL>
<LI><CODE>(sed | awk) ! perl & regexp</CODE>
<LI><CODE>(sed or awk) not perl and regexp</CODE>
</UL>

<P>
のように指定します。
</P>

<P>
といった検索をすることができます。
これは「 sed または awk が含まれ、perl は含まれない、そして regexp が含まれる」文書を検索するという意味になります。
括弧のネストもできるので、さらに複雑な検索式で検索することも可能です。
</P>

<P> フレーズをダブルクォーテーションまたは中括弧 '{' '}' で囲むこと
でフレーズ検索ができます。ただし、精度は 100% ではありません。とき
どきはずれます。たとえば </P>

<UL>
<LI><CODE>"SIMPLE MAIL TRANSFER PROTOCOL"</CODE>
<LI><CODE>{SIMPLE MAIL TRANSFER PROTOCOL}</CODE>
</UL>

<P>
のように指定します。
</P>

<HR>
<P>
この全文検索システムは
<STRONG><A HREF="http://www.ring.gr.jp/openlab/namazu/" >Namazu</A> v1.3.0.6</STRONG>
によって構築されています。
</P>
</BODY>
</HTML>

Index ファイルを作成するためのファイル copyy2

```
#!/bin/awk −f


{line[NR] =$3
 line2[NR] = $8
 line3[NR] = $11
}


END{
    for (i = 0; NR > i; i = i + 1){
        if(line2[i] ∼ /html$/)
        if(line[i] !∼ /ˆF/){
          a = substr(line[i],3,2)
          b = substr(line[i],5,2)

        #キーワード検索結果のローカルアドレスから URL へ変換する。
      print "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"0;URL="line3[i]"\">" > ENVIRON["index"]line[i]".html"
        #Web ページの内容をコピーする。
      print "tail +7 " ENVIRON["cache"]a"/"b"/"line[i] " >> "ENVIRON["index2"]line[i]".html"
    }
    }
}
```

Index 作成ファイル mkindex

```
#!/bin/sh

. /home/takatoku/bin/settei.conf

cd $pro2
$log"copyy2" $store > $log"all"
all
cd $index2
mknmz $index2
```

Namazu の設定ファイル .namazurc

This is the Namazu configuration file.
originally, this file is named 'namazu.conf-dist'. so you should
copy this to 'namazu.conf' to use.
#
item and value are MUST be separated with TAB character.
see "manual.html#NAMAZU_CONF" for detailed information.
<Directive List>
* INDEX   : Pathname where index file (NMZ.*) is placed.
* REPLACE : Replace URL string for search result output.
describe by TARGET, REPLACEMENT order.
if you do not want to do this replacement in command
line use, you can run 'namazu' with −U option and
avoid this processing.
* BASE   : append <BASE HREF="..."> to search result HTML.
this value must terminate with '/' or '\' character.#   * LOGGING : set OFF to turn off search keyword logging.
default: logging ON (to NMZ.slog)
* LANG   : set language code registrated in ISO 639
such as 'ja', 'en', 'de', and etc.#   if you set 'de' to this, namazu would use
NMZ.(head|foot|body|msg).de as message files.
* SCORING : set scoring method TFIDF or SIMPLE.
#
#Namazu が利用するインデックスを置くディレクトリを指定する。
INDEX /home/takatoku/public_html/index/
#検索結果を表示する URL を変更する。
REPLACE /home/takatoku/public_html/index/ http://*www.fken/∼takatoku/index/*
#相対パスで表される URL の基準点を設定する。
BASE http://*www.fken/∼takatoku/*
LOGGING ON
LANG ja
SCORING TFIDF

検索機能を付けた個人別履歴
インストールマニュアル
インストール手順

以下のファイルをそれぞれのディレクトリに保存してください。

- /usr/local/bin/

log3	store.log と cache からタイトル・内容を取得するファイル
log4	access.log からユーザ情報を取得し、一日分の全ユーザ履歴を作成するファイル
log13	log3・log4 を実行させるファイル
log14	作成された履歴をバックアップさせるファイル
log15	保存されているキーワード検索用ファイルのバックアップと Index 作成
settei.conf	設定ファイル
copyy2	Index ファイルを作成するためのファイル
mkindex	Index 作成ファイル

- /public_html/cgi-bin/

usr0.cgi	一日分の全ユーザ履歴から個人別履歴を作成するファイル
usr1.cgi	過去の個人別履歴を表示させるファイル
usr2.cgi	〃
usr3.cgi	〃
usr4.cgi	〃
usr5.cgi	〃
usr6.cgi	〃
.namazurc	Namazu の設定ファイル
search.html	キーワード検索のメニューページ
namazu.cgi	キーワード検索を行うファイル

以下のファイルのパーミッションをそれぞれ変更してください。

log3	755
log4	755
log13	755
log14	755
copyy2	755
mkindex	755
usr*.cgi	755
.namazurc	644

```
serch.html  644
namazu.cgi  755
```

それぞれのファイルを編集してください。

インストールファイルの編集

- settei.conf

  設定ファイルには、個人別履歴の作成に必用な Squid にあるファイルや保存するファイルのパスを設定します。

```
export cache="/var/spool/squid/"                    squid のキャッシュディレクトリがあるパス
export access="/var/log/squid/access.log"           squid の access.log がある絶対パス
export store="/var/log/squid/store.log"             squid の store.log がある絶対パス

export log="/usr/local/bin/"                        実行ファイルがある絶対パス
export pro1="/home/takatoku/public_html/"           process1・process2 ファイルが保存されているディレクトリがある絶対パス
export pro2="/home/takatoku/public_html/today/"     当日の process1・process2 ファイルがある絶対パス
export index1="/home/takatoku/public_html/index/"   Index ファイルが保存されているディレクトリがある絶対パス
export index2="/home/takatoku/public_html/index/today/"   当日の Index ファイルがある絶対パス
```

- log13、log14、mkindex

```
. /home/takatoku/bin/settei.conf                    settei.conf がある絶対パス
```

- .namazurc

  このファイルの設定はキーワード検索の結果表示を変更するためのものです。

  INDEX /home/takatoku/public_html/index/

  とあるところは、Index のあるディレクトリのパスを指定するところです。環境に合わせて変更してください。

  REPLACE /home/takatoku/public_html/index/ http://www.fken/~takatoku/index/

  とあるところは、Index が内部の絶対パスで始まる URL を検索結果として返すので、これを変換するためのものです。以下にその例をあげます。

```
file:/home/takatoku/public_html/index/index.html
                    ↓
http://www.fken/~takatoku/index/index.html
```

各空白 (REPLACE と/home/takatoku/public_html/index/との空白) は、「スペース」ではなく「Tab」をもちいて行ってください。

- search.html

  フォームの中で、

  <FORM METHOD=”GET” ACTION=”http://www.fken/~takatoku/cgi-bin/namazu.cgi”>

  とあるところを、namazu.cgi が置かれているパスに設定してください。

  フォームの中で、

```
<STRONG>検索対象</STRONG><BR>
<UL>
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="today" CHECKED>今日
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="1" CHECKE>1 日前
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="2" CHECKE>2 日前
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="3" CHECKE>3 日前
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="4" CHECKE>4 日前
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="5" CHECKE>5 日前
<LI><INPUT TYPE="CHECKBOX" NAME="dbname" VALUE="6" CHECKE>6 日前
</UL>
```

  とあるところは、検索対象の指定をする所で、

  VALUE=”today”

  は、検索するディレクトリ名を指定しています。ディレクトリ名を変更した場合はこちらの設定を変更してください。また検索するディレクトリを増やしたりした場合もこちらを変更してください。

  その他の部分のカスタマイズは好みに応じて変更していただいて結構です。

個人別履歴に必要な IDENT_DAEMON の設定を行います。

- /etc/httpd/conf/http.conf

  httpd.conf の最後の行に、IdentityCheck on　を付け加えて httpd を再起動してください。

- /etc/squid/squid.conf

```
#  TAG: ident_lookup_access
#       A list of ACL elements which, if matched, cause an ident
#       (RFC 931) lookup to be performed for this request.  For
#       example, you might choose to always perform ident lookups
#       for your main multi-user Unix boxes, but not for your Macs
#       and PCs.  By default, ident lookups are not performed for
#       any requests.
```

```
#
#       To enable ident lookups for specific client addresses, you
#       can follow this example:
#
#       acl ident_aware_hosts src 198.168.1.0/255.255.255.0
#       ident_lookup_access allow ident_aware_hosts
#       ident_lookup_access deny all
#
#       This option may be disabled by using --disable-ident with
#       the configure script.
#ident_lookup_access deny all
```

次の行に、ident_lookup_access allow all を付け加えて再起動してください。

クライアントマシンに IDENT_DAEMON をインストールしてください。

1. まず以下のファイルをダウンロードしてください。

   http://www.acs.ucalgary.ca/m̃mastrac/downloads/identd15.zip

2. 次に先ほどダウンロードした identd15.zip を展開し、展開されたファイルを

   ```
   C:\Program Files\identd\
   ```

   にコピーします。

   さらに Identd.exe のショートカットを作成し、プロパティのリンク先を

   ```
   "C:\Program Files\identd\identd.exe" /SERVICE /NOSHUTDOWN /READONLY
   ```

   に変更してください。

3. 最後に Identd.exe へのショートカットを実行してマシンを再起動してください。これでマシンを起動するたび自動的に IDENT_DAEMON が実行されます。

crontab の設定をします。

crontab を使用して個人別履歴を作成するには log13 をだいたい 20 ~ 30 分おきに実行させてください。ただし、この時間は実行させる環境によって左右されます。log14 は日付が変わる時間の前後に行ってください。mkindex に関しては処理に時間がかかりますので３０分以上で行ってください。

```
(例)*/30  9-21  *  *  * /home/takatoku/bin/log13
    30  23  *  *  * /home/takatoku/bin/log14
    */30  9-21  *  *  *  /home/takatoku/bin/mkindex > /dev/null
    30  23  *  *  * /home/takatoku/bin/log15
```

URL にリンクをはる。

最後に個人別履歴・キーワード検索が実行できるように URL のリンクを usr.cgi・search.html にはってください。います。

検索機能を付けた個人別履歴
運用マニュアル

個人別履歴を作成するために log13・log14 を実行させます。

log13 を実行すると個人別履歴の作成に必要なファイルである process1 と process2 が作成されます。これらのファイルは個人別履歴の作成に必要ですので削除しないでください。

log14 は個人別履歴の作成に必要なファイルをバックアップさせるためのファイルです。デフォルトでは当日を含めて一週間分の履歴を見ることができるようにしていますが、log14 の設定を変更していただければより長く履歴を保存することができます。

キーワード検索を行うために mkindex を実行させます。

mkindex を実行させるとキーワード検索の対象となる Web ページが設定したディレクトリに保存されます。保存されるファイルは全て合わせますと、非常に多くのファイル数とデータ量になりますので注意してください。

次に全文検索を行うための Index ファイルが作成されます。この作業にはマシンに非常に大きな負荷と、時間がかかりますので注意してください。

mkindex を実行させると NMZ.*のファイルが作成されます。これらのファイルは Index としての役割を果たすので削除しないでください。特に以下のファイルは重要なのでファイルの削除はもちらん変更もしないでください。

NMZ.i　NMZ.ii　NMZ.f　NMZ.fi　NMZ.h　NMZ.r

copyy2

本研究のキーワード検索は、キーワード検索の対象となる Web ページを指定したディレクトリに保存します。そのため検索結果から web ページを見るとその保存した Web ページを表示します。その後、

print ”tail +7 ” ENVIRON[”cache”]a”/”b”/”line[i] ” >> ”ENVIRON[”index”]line[i]”.html”

によって、Web ページの URL に自動的に飛ぶようになっています。これによってユーザは自分が過去に見た Web ページをいったん見て、その後最新の Web ページを見ることができるようになっています。Web ページの URL に自動的に飛ぶタイミングは CONTENT=0̈ とあるところの最後の 0 を変更してください。この設定では 0 秒後に Web ページの URL に自動的に飛ぶようになっています。