URL の履歴の自動作成

96H716 宮北哲博

目 次

1 はじめに **1**

1.1 現在の URL の履歴 1

1.2 本研究の URL の履歴 2

1.3 Proxy サーバのキャッシュ機能 2

2 URL の履歴 Web ページ **3**

2.1 ident デーモン 4

2.2 Proxy サーバの設定 4

2.3 HTTP サーバの設定 5

2.4 Web ページの情報の表示 5

3 キャッシュWeb ページ **7**

3.1 キャッシュの方法 7

3.2 指定日時に実行する方法 7

4 考察 **8**

5 まとめ **9**

1　はじめに

近年、Web ページでさまざまな情報を閲覧することができる。大学の演習室や研究室でも Web ページを閲覧する機会が増えている。そこで過去に閲覧した Web ページの一覧を見たいときがある。WWW ブラウザにも過去に訪れた Web ページの履歴の一覧を表示する機能があるが、これは大学の演習室や研究室のような複数のパーソナルコンピュータを不特定多数の人が共用する環境では、自分専用の過去の履歴を見ることができない。

そこで Proxy サーバのログファイルに Web ページを閲覧した時の URL などの記録が残されているので、それを用い自動的に URL の履歴を Proxy サーバ上で作成し、WWW ブラウザで閲覧できるようにする。

さらに、快適に Web ページを閲覧するための付加機能として、学生に授業などで Web ページを資料として閲覧させるときに、事前に Proxy サーバのキャッシュディレクトリに Web ページのデータをキャッシュしておけば高速に Web ページが表示されるようになるので、教員が Web ページ上で URL と日時を指定して Proxy サーバのキャッシュディレクトリにキャッシュできるものを作成する。

1.1　現在の URL の履歴

図 1のように WWW ブラウザに過去に訪れた Web ページの履歴の一覧を表示する機能がある。

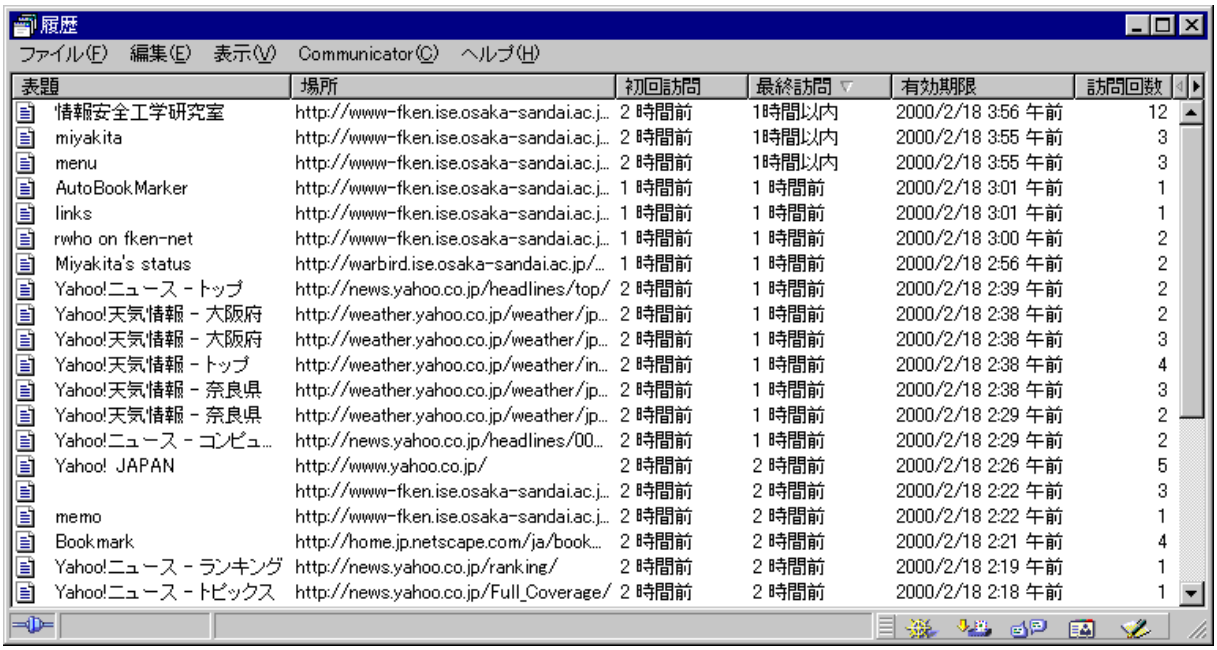


図 1: Netscape Navigator の履歴の表示

しかし、大学の演習室や研究室のような環境において、WWW ブラウザの履歴の表示では次のような問題点がある。

- WWW ブラウザの履歴データが保存されている先は、使用したクライアントマシンなので、次に履歴を表示させる場合には、同じクライアントマシンを使用しなければならない。
- 多数の人がクライアントマシンを使用するので、他人の履歴が混ざってしまって自分専用の履歴を見ることができない。

このような理由で、WWW ブラウザの履歴の表示はユーザにあまり使われない機能になってしまう。

1.2 本研究の URL の履歴

本研究では現在の URL の履歴の問題点を解決し次のような利点がある。

- どのクライアントマシンを利用しても URL の履歴が閲覧できる。
- 個人専用の URL の履歴が閲覧できる。
- 異なる種類の WWW ブラウザの使用に関係なく URL の履歴が残る。
- URL の表示だけでなく、Web ページの内容などが表示される。

1.3 Proxy サーバのキャッシュ機能

キャッシュ機能とは、一度取り寄せた Web ページを Proxy サーバが管理するディスク領域に保管しておくことである。これにより複数のクライアントが同じ外部の Web ページを参照した場合に一度取り寄せたものを使うことでネットワーク帯域を効率よく使用し、またクライアントの応答も改善できる。

このキャッシュ機能を利用し、URL と日時を指定して Proxy サーバのキャッシュディレクトリにキャッシュできるものを作成する。

2　URL の履歴 Web ページ

URL の履歴をクライアントマシンを使っているユーザ別に閲覧できるようにするためには、Proxy サーバに残っているログファイルだけでは、誰がどの Web ページを閲覧したか分からないので、Proxy サーバの設定を ident デーモンを利用する設定に変更し、ログファイルにユーザ名が記録されるようにする。また、URL の履歴をユーザごとに閲覧できるようにするために、この場合も ident デーモンでユーザ名を確認して、そのユーザの URL の履歴だけを表示する。

そして、URL だけでは情報が少ないので、Web ページのタイトル、内容、最終更新日を表示する。

図 2に表示結果の例を示す。

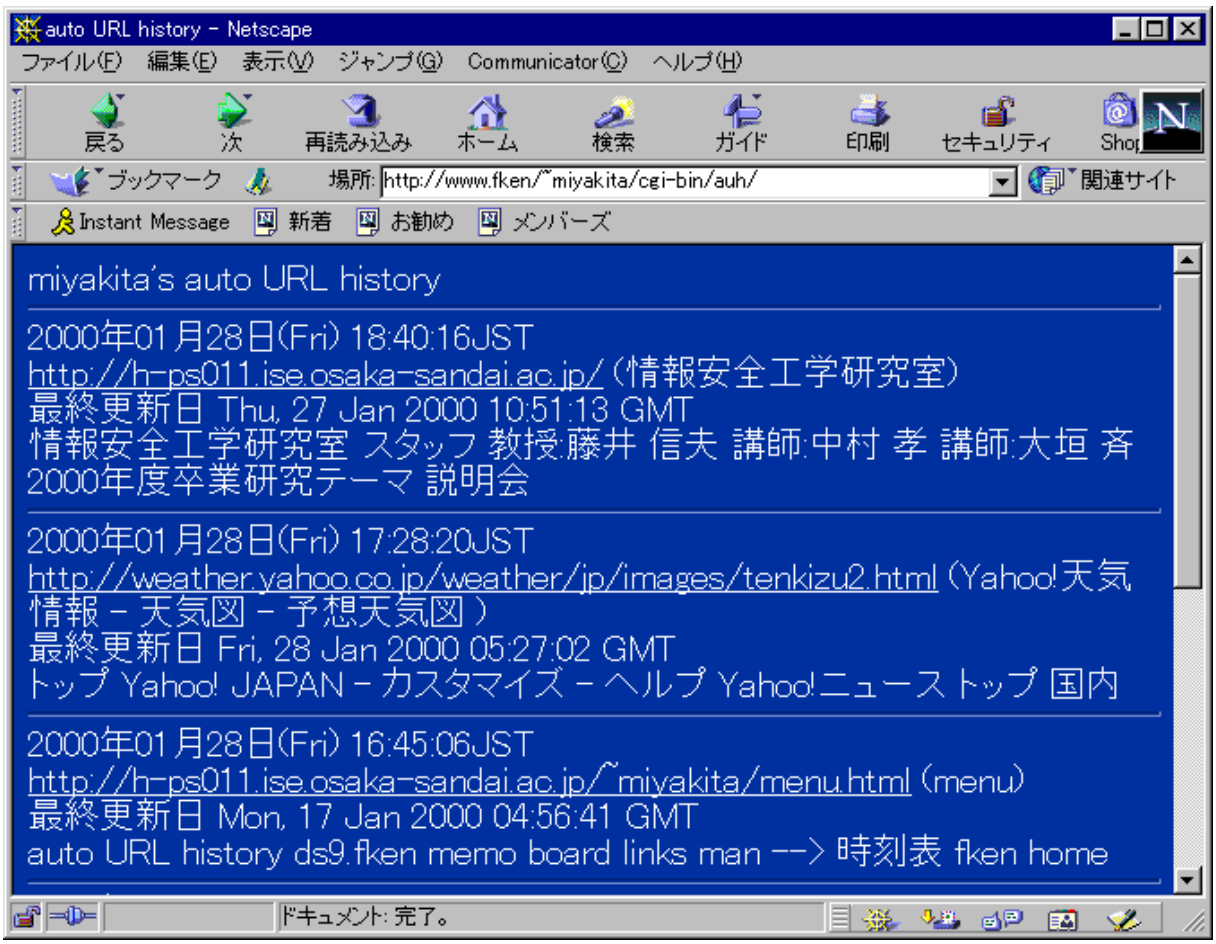
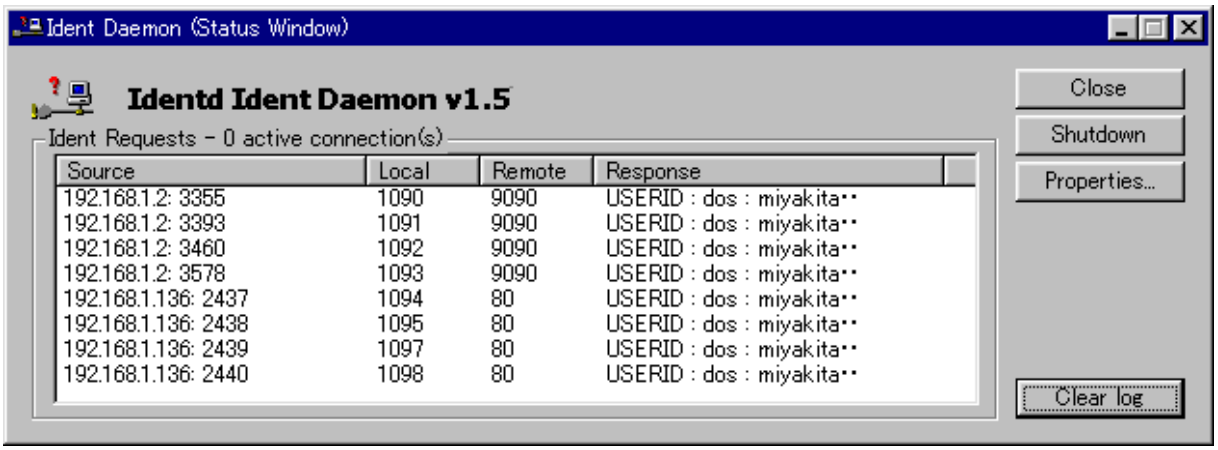


図 2: URL の履歴 Web ページ

2.1 ident デーモン

ident デーモンとは、Proxy サーバなどに接続しているクライアントのユーザ名を返してくるデーモンである。図 3に Windows の ident デーモンのステータスウインドウを示す。



Source	Local	Remote	Response
192.168.1.2: 3355	1090	9090	USERID : dos : miyakita･･
192.168.1.2: 3393	1091	9090	USERID : dos : miyakita･･
192.168.1.2: 3460	1092	9090	USERID : dos : miyakita･･
192.168.1.2: 3578	1093	9090	USERID : dos : miyakita･･
192.168.1.136: 2437	1094	80	USERID : dos : miyakita･･
192.168.1.136: 2438	1095	80	USERID : dos : miyakita･･
192.168.1.136: 2439	1097	80	USERID : dos : miyakita･･
192.168.1.136: 2440	1098	80	USERID : dos : miyakita･･

図 3: ident デーモンのステータスウインドウ

2.2 Proxy サーバの設定

ident デーモンを利用するために、/etc/squid/squid.conf を以下のように設定し、Proxy サーバのログファイルにユーザ名が記録されるようにする。

```
#  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
ident_lookup_access allow all
```

この設定を有効にすると、以下のようにログファイル (/var/log/squid/acsses.log) にユーザ名が入る。

```
947227966.393 5 192.168.1.81 TCP_MISS/200 298 HEAD http://www.zdnet.co.jp/ miyakita
DIRECT/www.zdnet.co.jp text/html
```

これを用いて、URL の履歴をユーザごとに作成する。

2.3 HTTP サーバの設定

ident デーモンを利用するために、/etc/httpd/conf/httpd.conf に
IdentityCheck on
を追加し、設定を有効にすると、REMOTE_IDENT 環境変数に、接続ユーザ名が入る。REMOTE_IDENT 環境変数を利用して、特定のユーザだけの URL の履歴を表示させる。

2.4 Web ページの情報の表示

Web ページのタイトル、内容、最終更新日を表示するために、Proxy サーバのキャッシュディレクトリにキャッシュされているものを用い、情報を取り出す。まず、Proxy サーバのログファイル (/var/log/squid/store.log) に、URL とその URL の Web ページのデータのキャッシュされているファイル名が記録されているので、そこから必要なファイルを探し出す。

次にキャッシュされているデータから Web ページのタイトル、内容、最終更新日を取り出す。

以下にキャッシュされているデータの例を示す。

```
http://www-fken.ise.osaka-sandai.ac.jp/index.html HTTP/1.0 200 OK
Date: Sat, 05 Feb 2000 09:46:15 GMT
Server: Apache/1.3.6 (Unix)  (Red Hat/Linux)
Last-Modified: Mon, 31 Jan 2000 07:53:32 GMT
ETag: "70078810-969-38953f7c"
Accept-Ranges: bytes
Content-Length: 2409
Content-Type: text/html
Age: 0
X-Cache: MISS from TrokNor.osaka-sandai.ac.jp
Proxy-Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>情報安全工学研究室</title>
<link rel="stylesheet" type="text/css" href="http://www.ise.osaka-sandai.ac.jp/%7Eandrew/andrew.css">
</head>

<body>
<h1>情報安全工学研究室</h1>

<ul>
<li>スタッフ
 <ul>
 <li>教授:藤井 信夫
 <li>講師:<a href="http://www.ise.osaka-sandai.ac.jp/~nakamura/"> 中村 孝</a>
 <li>講師:<a href="http://www.ise.osaka-sandai.ac.jp/~andrew/"> 大垣 斉</a>
 </ul></li>
```

```
<li>2000 年度卒業研究テーマ
<a href="/fken2000/fken2000tm.html"> 説明会資料</a>

<li><a href="/fken99g/">1999 年度卒業研究テーマ</a>
<a href="/fken99g/fken99tm.html">( 募集時の予定)</a>

<li><a href="/fken98g/">1998 年度卒業研究テーマ</a>
<a href="/fken98g/fken98tm.html">( 募集時の予定)</a>

<li><a href="./fken95-97g/">卒業研究テーマ (1995 ~ 1997)</a>
・
・(中略)
・
<hr>
<a href="mailto: andrew@osaka-sandai.ac.jp">andrew@osaka-sandai.ac.jp</a>
<br>

</body>
</html>
```

タイトルは、「<title>情報安全工学研究室</title>」のような、タイトルタグで囲まれているところから取り出す。

内容は、ボディタグ (<body>) 以降から、html タグを取り除いたテキストデータを取り出す。

最終更新日は、「Last-Modified: Mon, 31 Jan 2000 07:53:32 GMT」というのが最終更新日なので、ここから取り出す。

3　キャッシュWeb ページ

キャッシュWeb ページとは、図 4のように WWW ブラウザから、URL と日時を入力し、指定した日時で、指定した URL の Web ページのデータを、Proxy サーバのキャッシュディレクトリにキャッシュできるものである。

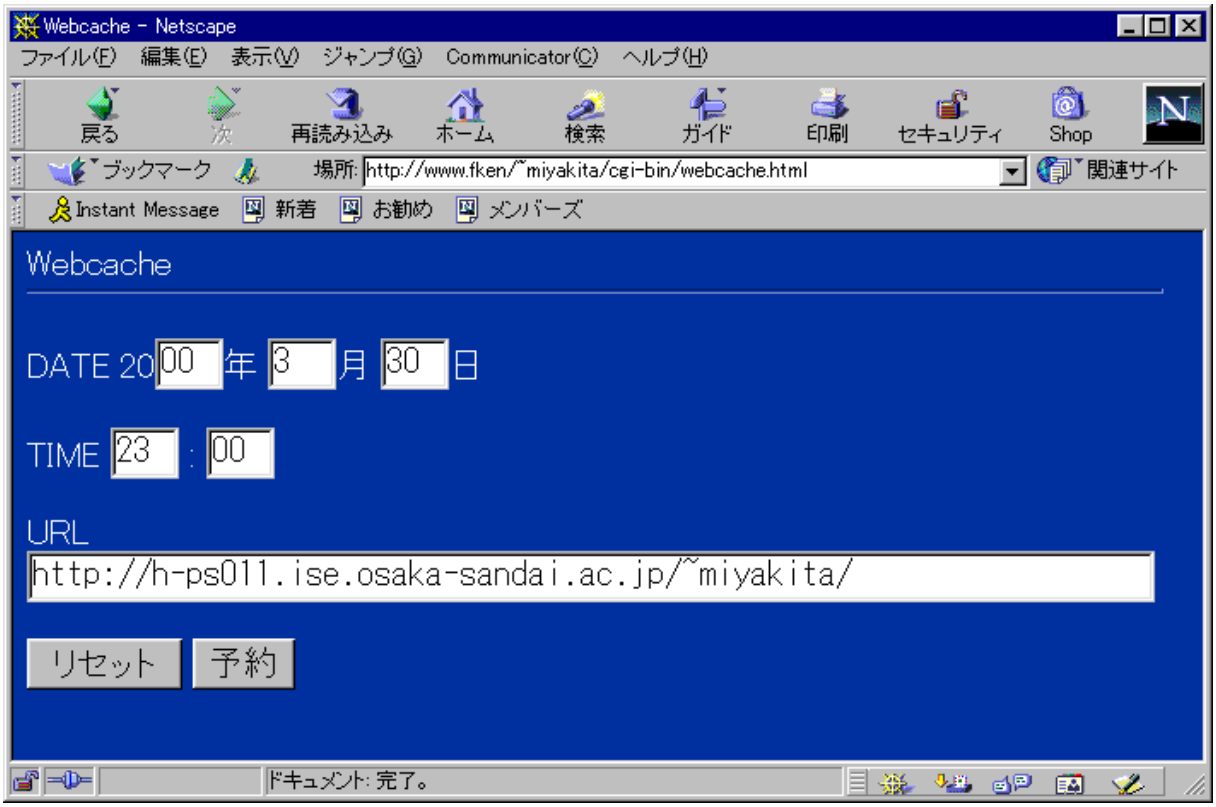


図 4: キャッシュWeb ページ

3.1　キャッシュの方法

Proxy サーバのキャッシュディレクトリにキャッシュする方法は、WWWcp という Web 自動巡回ソフトを使用して Proxy サーバ経由で Web ページのデータを取得して、Proxy サーバにキャッシュする。

WWWcp の特徴は、指定した URL のファイルを取得したあと、再帰的にファイルに含まれている URL を検索して、ファイルを取得する。その際、指定の URL のホスト、ポート、パス部分にマッチしない URL は無視するため、木構造をなすようにファイルが配置される。このために、URL を一つ一つ指定しなくても Web サイトのページをまとめて取得することができる。

3.2　指定日時に実行する方法

指定された日時に実行するコマンドで、at コマンドがある。この at コマンドを用いて、WWWcp を実行させて Web ページのデータを取得する。

4　考察

本研究でクライアントマシンを使っているユーザごとに、URL の履歴を表示させることができた。しかし、URL の並んでいる順番が Web ページを閲覧した日時の新しい順に並んでいるものしかないので、ユーザが自由に URL 順、タイトル順、閲覧回数の多いもの順、更新されたものだけなどを表示できる機能などを付け加えれば使いやすくなると思われる。

Web キャッシュページでは、簡単に指定日時に、指定した URL の Web ページのデータを Proxy サーバのキャッシュディレクトリにキャッシュすることができた。しかし、指定した URL の Web ページにリンクが多くされている場合などは、リンク数に応じて、キャッシュする時間が増加する。また、同じ日時にキャッシュの予約が集中するとネットワークに負担がかかってしまう。

また、ユーザを認識するために ident デーモンを利用したが、これは Windows 系の OS では標準で実装されていないのでインストールする必要がある。

5　まとめ

本研究でクライアントマシンを使っているユーザごとに、Web ページの内容が分かりやすいように URL の履歴を表示させることができた。今後は、ユーザが指定した日数だけ履歴を保存する機能や、URL の履歴の機能と、Proxy のキャッシュディレクトリにキャッシュできる機能とを組み合わせて、URL の履歴で閲覧回数の多い URL は自動的に Proxy のキャッシュディレクトリにキャッシュする機能を追加すればさらに快適な Web ページの閲覧を実現できると思われる。

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

[1] UNIX コマンドポケットリファレンス 中西隆

[2] The GAWK Manual (http://infoshako.sk.tsukuba.ac.jp/ShakoDoc/GNU/AWK/jgawk-jman/html/)

[3] AWK を 256 倍使うための本 志村拓、鷲北賢、西村克信

[4] CGI&Perl ポケットリファレンス 藤田 郁、三島俊司

[5] WWWcp page (http://www.ff.iij4u.or.jp/~rewsirow/WWWcp/WWWcp.html)

付録

キャッシュから Web ページの情報を取り出すプログラム (title.awk)

```
#!/bin/awk −f

BEGIN{
#    cache = "/var/spool/squid/"
   cache = ENVIRON["cache"]
#    tmp = "/home/miyakita/bin/auh/log/auh.tmp"
   tmp = ENVIRON["log_path"]"auh.tmp"

   tail ="tail +2 "
   tr = " | head -2k | nkf -e | tr '\\n|\\r|\\t' ' '"
   IGNORECASE = 1
}

#メイン
#squid にキャッシュされているもので、URL が/、.html、.htm、.cgi、.aps、.txt、で終っているものを対象。
($11 ~ /\/$/ || $11 ~ /.html$/ || $11 ~ /.htm$/ || $11 ~ /.cgi$/ || $11 ~ /.asp$/|| $11 ~ /.txt$/) && ($3 !~ /FFFFFFFF/) && ($9 !~ /−/){
   time = $1
   file = cache substr($3, 3, 2) "/" substr($3, 5, 2) "/" $3
   size = $9
   url = $11

   title = http(file)

   printf "%s %s %s %s\t%s\n", time, size, file, url, title
}

#更新&タイトル&内容
func http(file,    last, text, text2, var, s, l, b, c, d)
{
#squid のキャッシュの 1 行目にバイナリデータがあるので、2 行目から 2k バイト分を取り出し、日本語コードを EUC に変換し、改行、タブをスペースに変換し、ファイルに書き出す。
   system(tail file tr " > " tmp)

#書き出したものを読み込む。
   if((getline var < tmp) == 1)
   {
#最終更新日の抜き出し。
      if((s = index(var, "Last-Modified: ")) > 0)
      {
         last = "最終更新日 "substr(var, s + 15, 29)
```

```
        }

#タイトルの抜き出し。
        if((s = index(var, "<title>")) > 0)
        {
            l = index(var, "</title>")
            if(l ≥ 1)
            {
                text = "(" substr(var, s + 7, l − s − 7) ")"
            }
        }

#Web ページの内容の抜き出し。html タグの除去。
        if((s = index(var, "<body")) > 0)
        {
            var = substr(var, s)
            while((index(var, "<")) > 0)
            {
                b = index(var, "<")
                c = substr(var, 1, b − 1)
                var = substr(var, b + 1)
                b = index(var, ">")
                d = substr(var, b+ 1)
                var = c d
            }
#連続するスペースは 1 つにまとめる。
            gsub(/  +/, " ", var)
#抜き出した内容を 100 バイトにする。
            text2 = substr(var, 1, 100)
        }
    }
    close(tmp)
    return text "<br>" last "<br>" text2
}

END{
#テンプファイルの削除。
    system("rm -f "tmp)
}
```

ユーザの閲覧した URL と Web ページの情報を組み合わせるプログラム (usr.awk)

```
#!/bin/awk −f

BEGIN{
#    title_log = "/home/miyakita/bin/auh/log/title.log"
   title_log = ENVIRON["log_path"]"title.log"

   IGNORECASE = 1
}

#メイン
#ユーザ名が記録されているログで、URL が/、.html、.htm、.cgi、.aps、.txt、で終っているものを対象。
($7 ~ /\/$/ || $7 ~ /.html$/ || $7 ~ /.htm$/ || $7 ~ /.cgi$/ || $7 ~ /.asp$/ || $7 ~ /.txt$/) && ($7 !~
/&/) && ($8 !~ /−/){
   time = $1
   time2 = strftime("%Y 年%m 月%d 日 (%a) %H:%M:%S%Z", $1)
   ip = $3
   url = $7
   usr = $8
   title = ftitle(url)
   printf "%s %s %s %s %s\t%s\n", time, usr, ip, time2, url, title
}

func ftitle(url,    var, txt, sp)
{
   grep = "grep ' "url"\t' "title_log

#title.awk で処理したファイルから、同じ URL を検索し、Web ページの内容を取り出す。
   if((grep | getline var) == 1)
   {
      tab = index(var, "\t")
      var = substr(var, tab + 1)
      txt = var
   }
   close(grep)

   return txt
}
```

usr.awk を crontab で 5 分ごとに動作させるプログラム (usr-day.sh)

```
#!/bin/sh

#設定ファイルの読み込み
#. /etc/auh/auh.conf
. /home/miyakita/bin/auh/auh.conf

#usr.awk で$access_log をフィルタ処理し、最新順にソートし、テンポラリファイルに書き出す。
$auh_path"usr.awk" $access_log | sort -r > $log_path"usr.tmp"

#テンプファイルから、usr.log にリネーム。
mv -f $log_path"usr.tmp" $log_path"usr.log"
```

usr.awk を crontab で 1 日ごとに動作させるプログラム (usr-week.sh)

```
#!/bin/sh

#設定ファイルの読み込み
#. /etc/auh/auh.conf
. /home/miyakita/bin/auh/auh.conf

#ログファイルの入れ換え。
mv -f $log_path"usr.log.6" $log_path"usr.log.7"
mv -f $log_path"usr.log.5" $log_path"usr.log.6"
mv -f $log_path"usr.log.4" $log_path"usr.log.5"
mv -f $log_path"usr.log.3" $log_path"usr.log.4"
mv -f $log_path"usr.log.2" $log_path"usr.log.3"
mv -f $log_path"usr.log.1" $log_path"usr.log.2"
mv -f $log_path"usr.log.0" $log_path"usr.log.1"

zcat $access_log".0.gz" | $auh_path"usr.awk" | sort -r > $log_path"usr.0.tmp"
mv -f $log_path"usr.0.tmp" $log_path"usr.log.0"
```

title.awk を crontab で 30 分ごとに動作させるプログラム (title-day.sh)

```
#!/bin/sh

#設定ファイルの読み込み
#. /etc/auh/auh.conf
. /home/miyakita/bin/auh/auh.conf

#title.awk で$store_log をフィルタ処理し、最新順にソートし、テンポラリファイルに書き出す。
$auh_path"title.awk" $store_log | sort −r > $log_path"title.tmp"

#テンプファイルから、title.log にリネーム。
mv $log_path"title.tmp" $log_path"title.log"
```

URL の履歴 Web ページ (index.html)

```
<html>
<head>
<title>auto URL history</title>
<meta http-equiv="Content-Type" content="text/html; charset=x-euc-jp">
</head>

<body bgcolor="#00319C" text="#FFFFFF" link="#FFFFFF" vlink="#FFFFFF" alink="#FF0000">
<font color="red" size="5">auto URL history</font>
<hr>
<font size="5">
<!--
今日の履歴の引数は usr.log
日数分の指定は、見たい日数分の −1。
-->
<blockquote>
<p><img src="./icons/text.xbm" alt="[txt]"> <a name="main" href="./html.cgi?usr.log">
今日 </a></p>
<p><img src="./icons/text.xbm" alt="[txt]"> <a name="main" href="./html.cgi?2">3 日分
</a></p>
<p><img src="./icons/text.xbm" alt="[txt]"> <a name="main" href="./html.cgi?7">8 日分
</a></p>

</body>
</html>
```

ユーザを識別し、URL の履歴を表示する CGI(html.cgi)

```
#!/bin/awk −f

BEGIN{
   usr_log = "/home/miyakita/bin/auh/log/usr.log"
   day = ENVIRON["QUERY_STRING"]
   usr = ENVIRON["REMOTE_IDENT"]

    print "Content-type: text/html\n\n"
   print "<html>\n<head>\n<title>auto URL history</title>\n"
   print "<meta http-equiv=Content-Type content=text/html; charset=x-euc-jp>\n"
   print "</head>\n"
  print "<body bgcolor=#00319C text=#FFFFFF link=#FFFFFF vlink=#FFFFFF alink=#FF0000>\n"
   printf "<font size=+1 color=red>%s\'s auto URL history</font><hr>\n", usr

#ユーザ識別&html 表示
#今日の履歴分の表示
   if(day == "usr.log")
   {
      while((getline < usr_log) > 0)
      {
         if($2 == usr)
         {
            tab = index($0, "\t")
            back = substr($0, tab + 1)
            printf "%s %s<br><a href=%s target=_top>%s</a> %s<hr>\n", $4, $5, $6, $6, back
         }
      }
      close(usr_txt)
   }
#指定日数分の表示
   else
   {
      for(i = 0; i ≤ day; i++)
      {
         usr_txt = usr_log "." i
         while((getline < usr_txt) > 0)
         {
               if($2 == usr)
               {
                  tab = index($0, "\t")
                  back = substr($0, tab + 1)
                 printf "%s %s<br><a href=%s target=_top>%s</a> %s<hr>\n", $4, $5, $6, $6, back
               }
         }
```

```
            close(usr_txt)
        }
    }
    print "\n</body>\n</html>"
}
```

キャッシュWeb ページ (webcache.html)

```
<html>
<head>
<title>Webcache</title>
<meta http−equiv="Content-Type" content="text/html; charset=x-euc-jp">
</head>

<body bgcolor="#00319C" text="#FFFFFF" link="#FFFFFF" vlink="#FFFFFF" alink="#FF0000">

<font color="red" size="5">Webcache</font>
<hr>
<font size="5">
<form action="./webcache.cgi" method="get">
DATE 20<input type="text" name="yy" size="2" value="00"> 年
<input type="text" name="mm" size="2" value="3"> 月
<input type="text" name="dd" size="2" value="30"> 日 <br>
<br>
TIME <input type="text" name="h" size="2" value="23"> :
<input type="text" name="m" size="2" value="00"><br>
<br>
URL <input type="text" name="url" size="64" value="http://h-ps011.ise.osaka-sandai.ac.jp/~miyakita/"><br>
<br>
<input type="reset" value="リセット">
<input type="submit" value="予約"><br>
</form>
</font>
</body>
</html>
```

キャッシュWeb ページの at コマンドを実行させる CGI プログラム (webcache.cgi)

```
#!/usr/bin/perl

#webcache の場所。
#$webcache = "/usr/local/bin/webcache";
$webcache = "/home/miyakita/bin/webcache";
#ログファイルのディレクトリ
$log = "/tmp/";

use 5.004;
use CGI qw(:cgi);

print qq(Content−type: text/html\n\n);
print qq(<html>\n<head>\n<title>webcache</title>\n);
print qq(<meta http−equiv=Content−Type content=text/html; charset=x−euc−jp>\n);
print qq(</head>\n);
print qq(<body bgcolor="#00319C" text="#FFFFFF" link="#FFFFFF" vlink="#FFFFFF" alink="#FF0000">\n);

my $url = param("url");
my $yy = param("yy");
my $mm = param("mm");
my $dd = param("dd");
my $h = param("h");
my $m = param("m");

$time = time;
#ログファイル名にユーザ名と IP アドレスと登録時間を使用
$log_path = "$log$ENV{REMOTE_IDENT}-$ENV{REMOTE_ADDR}-$time";

print qq(20$yy 年$mm 月$dd 日$h 時$m 分にキャッシュ開始。<br>URL →$url<br>\n);

#キャッシュする URL をファイルに書き出し、at コマンドで日時指定。
#ログファイル記録
system(qq(echo "$webcache $url  > /dev/null" > $log_path && at −f $log_path $h:$m $mm/$dd/$yy));

#ログファイル削除
#system(qq(echo "$webcache $url  > /dev/null && rm -f $log_path" > $log_path && at −f $log_path
$h:$m $mm/$dd/$yy));

#atq の表示
$_ = readpipe atq;
s/\n/<br>\n/g;
print qq($_);

print qq(\n</body>\n</html>);
```

WWWcp を Proxy サーバ経由で動作させるためのプログラム (webcache)

```
#!/bin/sh

#WWWcp の場所。
#export wwwcp_path="/usr/local/bin/WWWcp"
export wwwcp_path="/home/miyakita/bin/WWWcp"
#Proxy の場所。
export http_proxy="http://h-ps011.fken:9090/"

#テンポラリファイルのディレクトリ。
export tmp_dir="/tmp/webcache-$LOGNAME/"

#ディレクトリの作成。
mkdir -p $tmp_dir

#WWWcp の実行。
#-k Keep-Alive を使用
#-C プロキシサーバのキャッシュを無効化
$wwwcp_path -k -C $1 $tmp_dir

#テンポラリファイル削除
rm -rf $tmp_dir
```

おまけ-暴走した CGI プログラムを止めるための CGI プログラム (kill.cgi)

```
#!/usr/bin/perl

#kill −KILL

print "Content-type: text/html\n\n";
print "<html><head><title>kill -KILL</title>\n";
print "<meta http-equiv=Content-Type content=text/html; charset=x-euc-jp>\n";
print "</head>\n";
print "<body bgcolor=#00319C text=#FFFFFF link=#FFFFFF vlink=#FFFFFF alink=#FF0000>\n";

print "<form action=./kill.cgi method=get>\n";
print "PID<input type=text name=kill>\n";
print "<input type=submit value=kill 実行><br>\n";
print "</form>";
print "\n</body>\n</html>\n";
print "<plaintext>";

#指定された pid の終了。
($kill, $pid) = split(/=/, $ENV{QUERY_STRING});
if($pid > 0){kill KILL, $pid;}

#ps ux の表示。
print "USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND\n";
system "ps ux | grep '/home/' &";
```

auto URL history Version 0.1 β
インストールマニュアル

Copyright(C)1999-2000 宮北哲博

インストール環境

- Squid Cache: Version 2.2.STABLE4
- awk
- crond
- CGI 環境

ディレクトリの作成

- /etc/auh/
- /var/log/auh/

ファイルのコピー

以下のファイルを/etc/auh/にコピー

- auh.conf

以下のファイルを/usr/local/bin/にコピー

- title.awk
- usr.awk
- usr-day.sh
- usr-week.sh
- title-day.sh

以下のファイルを~/public_html/cgi-bin/にコピー

- index.html
- html.cgi

パーミッションの変更

- auh.conf (644)
- title.awk (755)
- usr.awk (755)

- usr-day.sh (755)
- usr-week.sh (755)
- title-day.sh (755)
- index.html (644)
- html.cgi (755)

設定ファイルの確認

auh.conf の内容

```
export cache="/var/spool/squid/"                squid のキャッシュディレクトリ
export access_log="/var/log/squid/access.log"   squid の access.log の絶対パス
export store_log="/var/log/squid/store.log"     squid の store.log の絶対パス

export auh_path="/usr/local/bin/"               auh インストールディレクトリ
export log_path="/var/log/auh/"                 auh ログディレクトリ
```

ident デーモンの設定 (Proxy サーバ)

/etc/squid/squid.conf を以下のように設定し、squid を再起動。

```
#  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
ident_lookup_access allow all
```

ident デーモンの設定 (HTTP サーバ)

/etc/httpd/conf/httpd.conf にの最後の行に

IdentityCheck on

を追加し、httpd を再起動。

クライアント (Windows 系) の ident デーモンのインストール

- ident デーモン (http://www.acs.ucalgary.ca/~mmastrac/downloads/identd15.zip)
- identd15.zip を解凍し、 `C:\Program Files\identd\` にコピー。
- identd のショートカットの作成。
- リンク先を
  `"C:\Program Files\identd\Identd.exe" /SERVICE /NOSHUTDOWN /READONLY`
  に設定。(図 5)

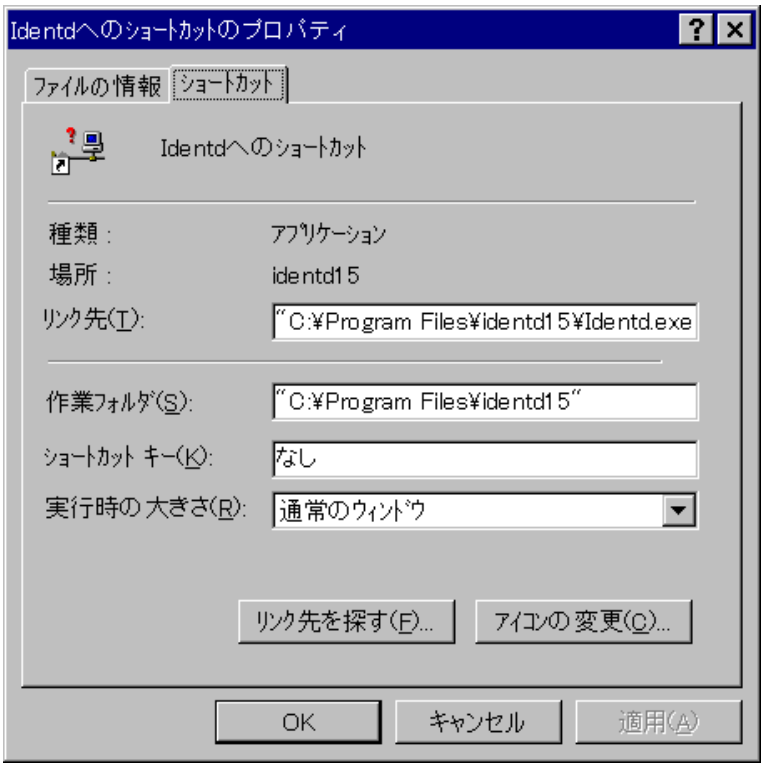


図 5: identd へのショートカットのプロパティ

- ショートカットをダブルクリック。
- マシンの再起動後、サービスとして ident デーモンが起動。

crontab の設定例

```
16,46 11-21 * * Mon-Sat /usr/local/bin/title-day.sh 2> /dev/null
*/10 11-21 * * Mon-Sat /usr/local/bin/usr-day.sh 2> /dev/null

0 2 * * * /usr/local/bin/usr-week.sh > /dev/null
```

auto URL history Version 0.1 β
運用マニュアル

Copyright(C)1999-2000 宮北哲博

注意事項

/var/log/auh/以下に生成されるファイルは履歴の表示に必要なので削除しないこと。

- title.log (Web ページの内容のログファイル)
- usr.log (今日の履歴のログファイル)
- usr.log.0 (1 日前の履歴のログファイル)
- usr.log.1 (2 日前の履歴のログファイル)
- usr.log.2 (3 日前の履歴のログファイル)
- usr.log.3 (4 日前の履歴のログファイル)
- usr.log.4 (5 日前の履歴のログファイル)
- usr.log.5 (6 日前の履歴のログファイル)
- usr.log.6 (7 日前の履歴のログファイル)
- usr.log.7 (8 日前の履歴のログファイル)

Webcache Version 0.1 β
インストールマニュアル

Copyright(C)1999-2000 宮北哲博

インストール環境

- perl5
- atd
- CGI 環境

ファイルのコピー

以下のファイルを/usr/local/bin/にコピー

- webcache
- WWWcp (http://www.ff.iij4u.or.jp/~rewsirow/WWWcp/WWWcp.html)

以下のファイルを~/public_html/cgi-bin/にコピー

- webcache.html
- webcache.cgi

パーミッションの変更

- webcache (755)
- WWWcp (755)
- webcache.html (644)
- webcache.cgi (755)

設定の確認

webcache の設定

```
export wwwcp_path="/usr/local/bin/WWWcp"          WWWcp の絶対パス
export http_proxy="http://h-ps011.fken:9090/"     Proxy サーバ

export tmp_dir="/tmp/webcache-$LOGNAME/"          作業ディレクトリ
```

webcache.cgi の設定

```
$webcache = "/usr/local/bin/webcache";            webcache の絶対パス
$log = "/tmp/";                                   ログファイルディレクトリ
```

Webcache Version 0.1 β
運用マニュアル

Copyright(C)1999-2000 宮北哲博

注意事項

webcache.cgi で変数$log で指定したディレクトリにログファイルが生成されるので削除しないこと。

- miyakita-192.168.1.81-951408765 (ログファイル名の例)

webcache で変数 tmp_dir で指定したディレクトリに、Web ページのキャッシュ中にキャッシュした Web ページのデータが生成されるが、キャッシュが終了すれば、自動的に削除される。

指定した URL の Web ページにリンクが多くされている場合などは、リンク数に応じて、キャッシュする時間が増加する。また、同じ日時にキャッシュの予約が集中するとネットワークに負担がかかる。