アクセスログを利用したアクセスパターンの解析

98H134 福井隆幹

目 次

1 はじめに **2**

2 アクセスログの解析 **3**
2.1 アクセスログの情報 3
2.2 アクセスログ解析の問題 4
2.3 アクセスログ解析の意義 6

3 アクセスパターン **7**
3.1 アクセスパターンの定義 7
3.2 アクセスパターンの利用方法 7

4 アクセスパターンの抽出 **8**
4.1 概要 8
4.2 解析前の設定 8
4.3 アクセスパターンの抽出 9
4.4 解析スクリプト 9
4.5 コマンドの入力 9
4.6 アクセスログ解析結果 10

5 まとめ **11**

1　はじめに

今日、学内ネットワークを構築している教育施設は多く情報化に伴い、学内ネットワークを利用し学生を対象とした教材としての Web コンテンツを作成する教育者もいる。Web コンテンツ製作者にとって、閲覧者が製作者 (ここでは教育者がコンテンツを製作すると考え、製作者と教育者を同一人物とする) の意図するアクセスの順番に沿って閲覧することは重要な問題である。例えば、簡単な問題から難しい問題へと製作者がアクセスしてもらうように考えていたとしても、閲覧者が逆から順に閲覧した場合、問題が難しいことによってページを読み進めることを途中で諦めてしまうことや、コンテンツの内容を理解できない可能性がある。

そこで、Web コンテンツ製作者は閲覧者がどのような順番でアクセスしているかを知り、製作者の意図しないアクセスが多い場合にはコンテンツを改良する必要がある。アクセスを調べる方法として、アクセス数を数えるアクセスカウンタがあるが、この場合には単純にアクセスされた回数のみを表示するためにどのようにアクセスしたかという順番までは解らない。しかし、Web コンテンツのアクセスの際に Web サーバに保存されるアクセスログを利用する方法がある。

アクセスログを解析することによってアクセスパターンを抽出することが出来る。抽出したパターンと Web コンテンツ製作者が考えているパターンとを比べる。その結果、Web コンテンツ製作者の意図通り閲覧されているか解る。意図通りでない場合には、抽出したパターンを参考にしてコンテンツの改良を行う必要がある。このように閲覧者を Web コンテンツ製作者の意図通りに導くために本研究ではアクセスログの解析手法を確立する。

2　アクセスログの解析

アクセスログとは Web コンテンツのアクセスの際に Web サーバに保存されるものである。アクセスログを取る主な理由として、不正アクセスがあった場合に不正アクセスをした人間を特定する手段として利用すること、アクセスログのデータを集計して閲覧状況やリンク元を調べることに利用することが考えられる。

2.1　アクセスログの情報

アクセスログは 1 つのファイルをリクエストするごとに以下のような情報が 1 行ずつ保存される。

```
fw2.osaka-sandai.ac.jp unknown - [29/Nov/2001:00:18:24 +0900]
"GET /home/andrew/work/mrtg-2.5.1/mrtg-m.png HTTP/1.0" 200 4707
"http://www.fken.ise.osaka-sandai.ac.jp/SNMP/omp-r.1.html"
"Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)"
```

この例では、表 (1) に示すような情報が読み取れる。

表 1: アクセスログに残る情報の一例

ページを閲覧するクライアントのホスト名または IP アドレス	fw2.osaka-sandai.ac.jp
ident	unknown
authuser	-
ページをリクエストした時間	[29/Nov/2001:00:18:24 +0900]
クライアントが行なったリクエスト	GET /home/andrew/work/mrtg-2.5.1/mrtg-m.png HTTP/1.0
リザルトコード	200
転送バイト数	4707
リファラ	http://www.fken.ise.osaka-sandai.ac.jp/SNMP/omp-r.1.html
クライアントのブラウザバージョン及び使用 OS	Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)

ページを閲覧するクライアントのホスト名または **IP** アドレス　クライアントがプロキシを介してアクセスしている場合にはプロキシのホスト名が入る。違うクライアントからのアクセスでも同じプロキシを利用している場合には同一のホストからのアクセスとなる。

ident　IdentityCheck が有効でクライアントの identd が動作しているときにユーザ名が返される。IdentityCheck を使用していない場合は '-' が書き込まれる。

authuser　ユーザ認証のアクセス制限をしていた場合、ユーザ名が書き込まれる。ユ-ザ認証していない場合には '-' が書き込まれる。

クライアントが行なったリクエスト　GET はリクエストの種類を示す。GET は HTML データや画像データなどを取得する場合や CGI にデータを渡す場合に使用する。HEAD はヘッダ情報だけが返され、ファイルサイズや更新日時を調べる場合に利用する。POST は主に CGI にデータを渡す場合に使用する。その後にくるパスはリクエストしたコンテンツが Web サーバのどこにおいてあるか表示され

る。 最後の HTTP/1.0 は HTTP のバージョンで、HTTP/1.0 および HTTP/1.1、HTTP プロトコルテストバージョンの HTTP/0.9 ある。

リザルトコード 100 番台が通信エラー、200 番台が正常終了、300 番台がリダイレクション、400 番台がクライアントエラー、500 番台がサーバエラーとなっている。代表的なものを表 (2) に示す。

リファラ リクエストしたコンテンツのパスの前に見た、つまりリンク元のページである。

表 2: 代表的なリザルトコードの意味

200	OK	送信が成功した時に返される
302	Moved Temporarily	新しい URL にリダイレクトされた時に返される
304	Not Modified	サーバがクライアントの要求に対し、クライアントのキャッシュを使用するよう命じた時に返される
403	Forbidden	アクセスが拒否された時に返される
404	Not Found	ユーザがアクセスしようとするページが存在しない時に返される
500	Internal Server Error	サーバエラーが発生した時に返される

2.2 アクセスログ解析の問題

webanalizer 図 **1**や analog 図 **2**のような従来のアクセスログ解析は主にアクセスの回数、ブラウザの種類、時間別アクセス数、アクセスした人のドメイン、リファラ (検索キーワード別など) である。これらの情報は月別や日別または曜日別ごとに HTML 形式で表示するものが多い。しかし、アクセスの遷移についての解析ソフトウェアは多くは存在しない。理由として、以下のような問題があると考えられる。

1. プロキシが使われていて同じクライアントのホストアドレスである場合には同一人物と見えてしまう。
2. Netscape や Internet Explorer のキャッシュ機能でキャッシュが読み込まれた場合には Web サーバに読み込みに来ない為、ログに残らない。
3. ISP でプロキシが使われていた場合、上記 2 と同じくキャッシュが読み込まれた場合にはログに残らない。
4. 閲覧者がページ A を閲覧した後、ページ B を閲覧し、別のサーバのページ X に移動した後にブラウザにある戻るボタンなどを利用し、ページ B に戻り、同じサーバにあるページ C に移動した場合、A → B → X → B → C となり、アクセスの遷移が途切れたと判断し A → B、B → C とするべきところが A → B → C とログに残ってしまい、アクセスの遷移が繋がっているように見えてしまう。
5. 閲覧者が複数ブラウザを立ち上げ、並行して同じサーバ内のページを閲覧した場合には正確なアクセスの遷移を得ることが出来なくなってしまう。

1の問題はクライアントのブラウザのバージョンや OS のバージョン、リファラ、閲覧した時間を利用してユーザ毎に分けることが出来るが、演習室のような環境である場合では、同じ OS、同じブラウザが使われることが多く、正確に分けることは困難である。

2や **3**の問題は「expires」や「no cache」のタグを入れることにより回避が出来る。

4の問題は深刻でページ B に移動したときリファラにはページ A のスが入っているため正確なアクセスの遷移が出すことが出来ない。

5の問題はリファラを利用することにより回避できる。

以上の問題からアクセス遷移を利用した解析ソフトが少ない理由はログだけでは正確なアクセスの遷移が取りづらいことが原因と考えられる。

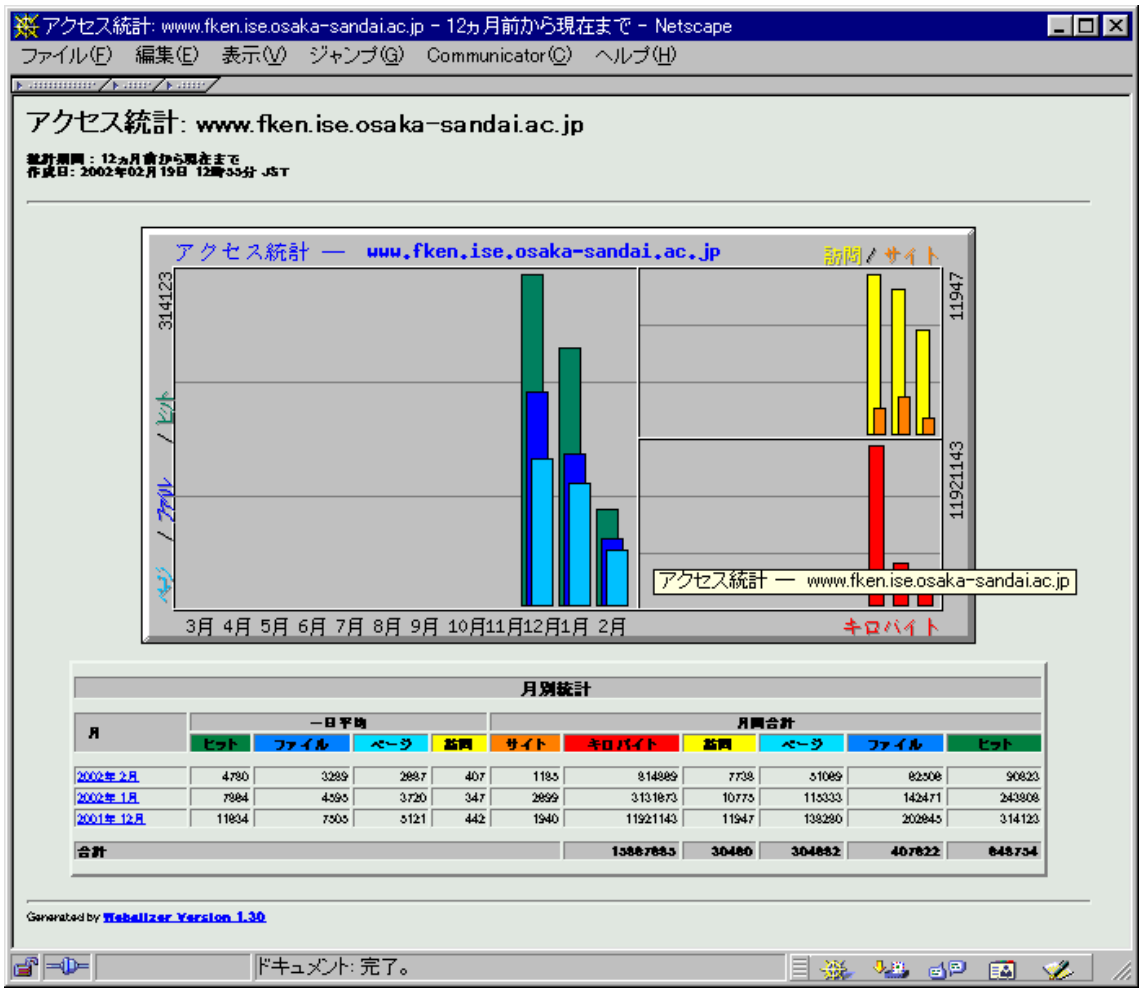

図 1: webanalizer の解析結果

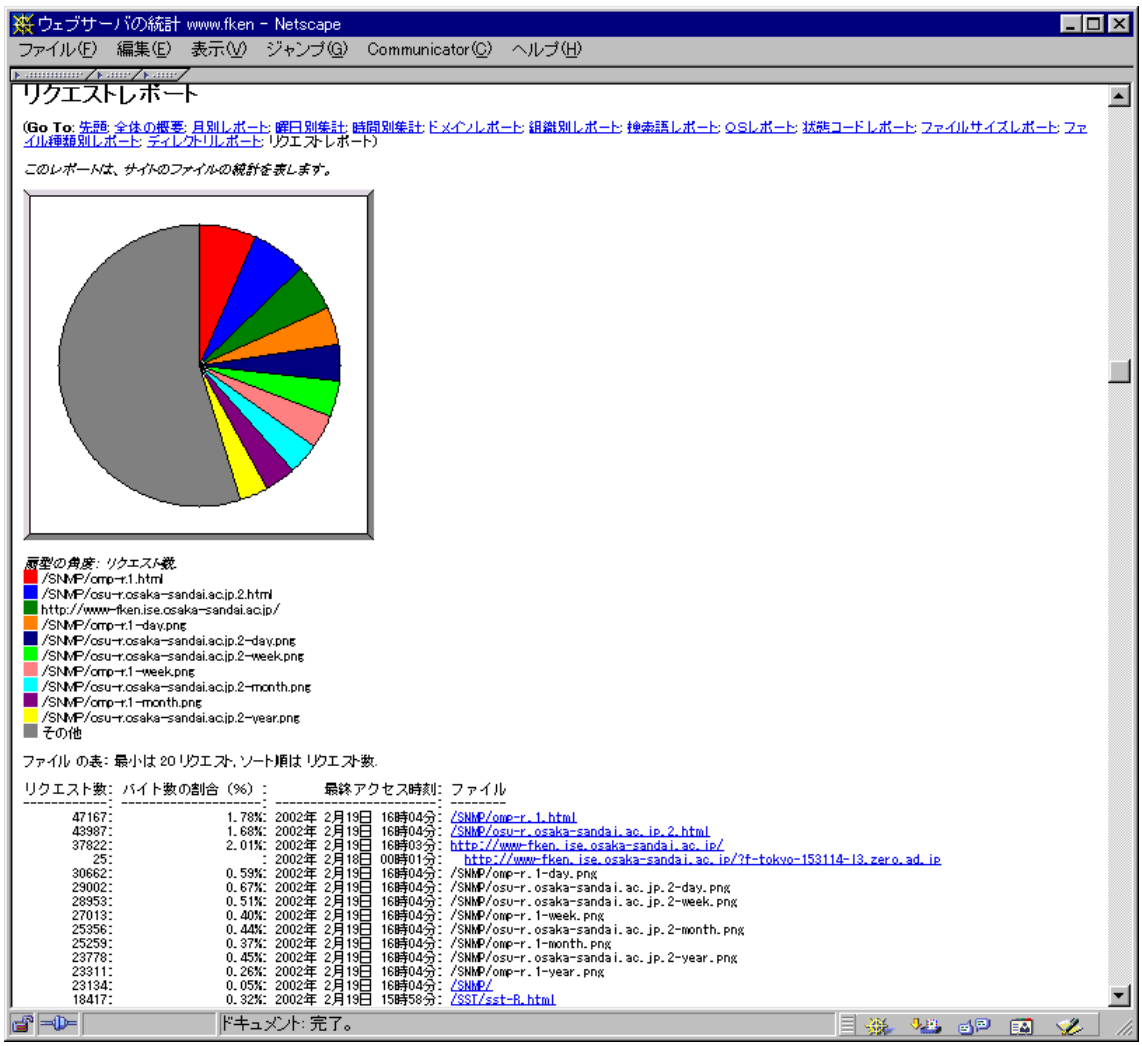

図 2: analog の解析結果

2.3 アクセスログ解析の意義

アクセスログを解析し正確なアクセスの遷移が取りづらいことは 2.2節で述べた。しかし、解析した結果が完全に正確なデータでは無くとも、参考になる程度の正確なデータであるなら解析した結果を利用してページを改良することは出来る。

3　アクセスパターン

この章では、本研究でのアクセスパターンとする場合の有効な閲覧時間やリンクの定義、アクセスパターンをどのように利用するか、また結果を出すことによってどのような利点があるかを説明する。

3.1　アクセスパターンの定義

アクセスパターンとは閲覧者が Web コンテンツを読み、そのコンテンツの中のリンクを辿って別のページに移動する行動が行なわれた時のパターンのことを指す。例えば、ページ A からページ B にリンクされていて、閲覧者がページ A からページ B にリンクを利用して移動した場合にはアクセスパターンは A → B とする。しかし、閲覧者が長時間ページを開いたまま一定時間以上 (ここでは仮に 3600 秒とする) 経った後のリンクは、ページを表示したまま長時間席を外していた、表示させたまま別のページを見ていた、と考えリンクが途切れたと判断し、本研究では 3600 秒以上経った後のリンクには別のアクセスパターンとして分けている。

多くのアクセスパターンを集計することにより、閲覧者がどのような順番でページを読み進み、どのページで読む進めることを止めたのかが解る。閲覧者のアクセスパターンのサンプルが増えてくれば集計を行ない、集計の結果により大抵の閲覧者が取る行動 (アクセスパターン) が見えてくる。

3.2　アクセスパターンの利用方法

アクセスパターンには 2 種類ある。図 (3) はページをアクセスしている様子である、円はページを表し、矢印が閲覧者のアクセスパターンを表している。一つ目は Web コンテンツ製作者の意図通りアクセスパターンを (A) とし、アクセスパターンの途中でアクセスパターンから外れてアクセスしたものを (B、C) とする。

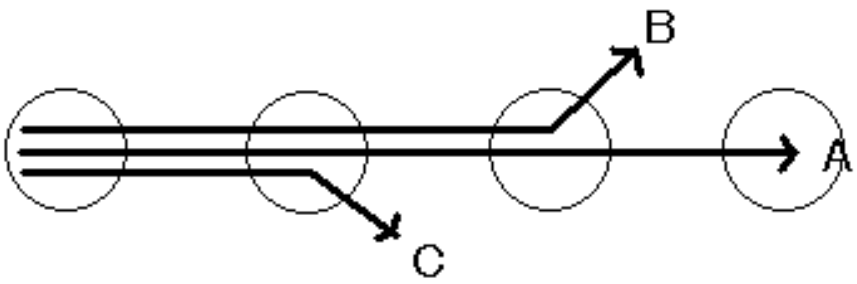

図 3: ページをアクセスしている様子

全体の割合は式 (1) で求められる。このとき、割合が大きければ、意図通りのアクセスをする閲覧者が多いことになるが、低くなるにつれてコンテンツのガイドが悪いということが解り、改良する目安となる。さらに、どのページに移動しているかを表示してページを改良する手助けになるデータを提供することによって、Web コンテンツ製作者の Web コンテンツ改良の際の負担軽減が期待される。

$$\frac{A}{A+B+C} \times \ 100(\%) \tag{1}$$

4　アクセスパターンの抽出

この章では本研究で確立した解析手法を用いた解析の方法を説明する。

4.1　概要

解析には Apache のアクセスログを利用する。スクリプトはアクセスログを読み込み、クライアントのホストアドレス、HTTP バージョン、ブラウザの種類、OS の種類で sort する。sort したログをアクセスパターンに分け、Web コンテンツ製作者が意図しているアクセスパターンと比較し、解析結果を HTML 形式のファイルに出力する。解析結果は HTML 形式であり、ブラウザ等で確認をする。

4.2　解析前の設定

Web コンテンツ及び、Web コンテンツ製作者が意図しているアクセスパターンを表記したファイル
図 (4)、アクセスログの 3 点である。

Web コンテンツ製作者が意図しているアクセスパターンを表記したファイルにはコンテンツを閲覧する順番で一行ずつ書き込む形式になっている。その際、アドレスをプロトコルからすべてを書く方法とプロトコルからドメイン名までの URL が http://www-fken.ise.osaka-sandai.ac.jp の場合にはディレクトリ以下のアドレスだけでも認識するようになっている。例えば、下の図 (4)、図 (5)、図 (6) は全て同じフォルダに置くようにしている。

```
/
http://www-fken.ise.osaka-sandai.ac.jp/%7Efukui/
http://www-fken.ise.osaka-sandai.ac.jp/%7Efukui/4su.html
/%7Efukui/index.cgi
```

図 4: Web コンテンツ製作者が意図しているアクセスパターンを表記したファイルの例 1

```
/
/%7Efukui/
/%7Efukui/4su.html
/%7Efukui/index.cgi
```

図 5: Web コンテンツ製作者が意図しているアクセスパターンを表記したファイルの例 2

```
http://www-fken.ise.osaka-sandai.ac.jp/
http://www-fken.ise.osaka-sandai.ac.jp/%7Efukui/
http://www-fken.ise.osaka-sandai.ac.jp/%7Efukui/4su.html
http://www-fken.ise.osaka-sandai.ac.jp/%7Efukui/index.cgi
```

図 6: Web コンテンツ製作者が意図しているアクセスパターンを表記したファイルの例 3

4.3　アクセスパターンの抽出

アクセスパターン抽出の方法として、まずアクセスログをクライアントのホストアドレスごとに分割する。分割したログを HTTP バージョン、ブラウザの種類、OS の種類が同じものを抜きだし、リファラがあるログをアクセス順に取り出したものをアクセスパターンとする。その際に閲覧時間が 3600 秒を越えたものはコンテンツを表示したまま別のコンテンツを見ていた、またはコンピュータから離れていたと考えてアクセスが途切れたとみなして、そこでアクセスパターンの終了とする。また、閲覧時間が 0 秒は有り得ないと考え、0 秒のものは無効とした。

4.4　解析スクリプト

一つのコマンドで解析が出来るようにしている。アクセスログを読み込み、クライアントのホストアドレス、HTTP バージョン、ブラウザの種類、OS の種類で sort し、sort したログをアクセスパターンに分け、アクセスパターンの合計数を取る。Web コンテンツ製作者が意図しているアクセスパターンと比較し、Web コンテンツ製作者が意図に沿っているアクセスパターンと意図と違うアクセスパターンに分け、それぞれの占める割合を解析結果として HTML 形式のファイルで出力する。スクリプト名は analyze.pl で付録に付けておく。

4.5　コマンドの入力

アクセスログから解析結果を表示するには以下の一つのコマンドで出来るようになっている。ただし、analyze.pl が実行可能になっていなければならない。実行時にはアクセスログが閲覧できるサーバにログインしておく必要がある。analyze.pl があるディレクトリで実行する場合は以下のコマンドとなる。

```
% ./analyze.pl
```

解析結果は main.pl で指定したディレクトリに保存することが出来る。結果を見るには Netscape 等ブラウザで見る。ここでは Netscape を使った場合のコマンドを載せる。

```
% netscape ${PATH}kekka.html
```

以上で、解析結果を見ることが出来る。

4.6　アクセスログ解析結果

解析結果は図 (7) のような HTML 形式のファイルで表示される。

図 7: 解析結果

結果に表示される内容は Web コンテンツ製作者の意図するアクセスパターンとそれに沿って閲覧者が閲覧した回数と全体に占める割合、その次に途中まで Web コンテンツ製作者の意図するアクセスパターンに沿っていた最後のページとそこから外れたページのみを表示し、その閲覧回数と全体に占める割合を表示した。それぞれの全体に占める割合は式 (1) によって求めた。

5　まとめ

本研究で従来のアクセス解析ソフトを利用し、Web コンテンツ製作者が行なってきた各ページの閲覧数だけで Web コンテンツを改良する方法より、アクセスパターンを提示することにより、どのページから Web コンテンツ製作者の意図と違うアクセスが多いか見ることが出来る。このことによって、Web コンテンツ製作者の負担が軽減されると期待できる。しかし、改良する際に推奨するパターンの提示やコンテンツの自動改良があれば、Web コンテンツ製作にとって更なる負担の軽減が期待できる。これについては今後の課題である。

また、本研究で確立した解析手法での問題として以下のようなものがある。

- サーチエンジン等のロボットを弾く有効な手段を取っていないためにロボットのアクセスパターンが混入する可能性が有り、信頼性が低い。
- 解析結果が文字情報のみで解りづらい。図示することによってもっと解りやすくなったのではないか。

これらの問題も今後の課題として改善すべきである。

謝辞

本研究を行うために、多大な御指導・御協力を頂いた大垣 斉講師、藤井 信夫教授、中村 孝講師、アドヴァイスをして下さった fken.a4w メーリングリストの方々、各研究生の方々に深く感謝の意を表します。

参考文献

[1] Larry Wall, Tom Christiansen, Randal L. Schwartz 共著, 近藤 嘉雪 訳, プログラミング Perl 改訂版, オライリー・ジャパン

[2] Cameron Newham, Bill Rosenblatt 共著, 近藤 美代子 訳, 入門 bash 第 2 版, オライリー・ジャパン

[3] 奥村 晴彦 著, 改訂版 LaTeX 美文書作成入門, 技術評論社

[4] 日本の Linux 情報 http://www.linux.or.jp/

[5] Google(検索サイト) http://www.google.co.jp/

付録

analize.pl スクリプト

————ここから————

```
#! /usr/bin/perl -w

#######
# 読み込むデータの例
# xxx.yyy.ap.so-net.ne.jp unknown - [22/Nov/2001:00:00:46 +0900] \
# "GET /fken2000/ohmae-mizusima/H/sld001.htm HTTP/1.1" 200 1176 \
# "http://www-fken.ise.osaka-sandai.ac.jp/fken2000/ohmae-mizusima/H/" \
# "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; DigExt)"
#######

my $LOGFILE = "/var/log2/httpd/access_log.0"; # 解析対象のアクセスログの PATH
my $LIST = "patern.dat"; # アクセスパターンを書いたファイルの場所
my $FILE = "/var/tmp/"; # tmp ファイル保存ディレクトリの場所
my $KEKKA = "/home/fukui/Perl/Dir/"; # 解析結果を保存する場所
my @main_uri_list;
my @access_patern;
my @list;
my @sortlog;

use strict;
{
    Patern();

#######
# 必要な部分を取り出す
# "$mon" を "%mon_list" で数値に換え、"$date" 順に sort できるようにする
#######
    my %mon_list =
qw(
   Jan 01
   Feb 02
   Mar 03
   Apr 04
   May 05
   Jun 06
   Jul 07
   Aug 08
   Sep 09
   Oct 10
```

```
   Nov 11
   Dec 12
   );

    my @log;
    my $uri;
    my $remainder;

    open(LOG, "< $LOGFILE") or die "$!\n";
    {
while(<LOG>)
{
    chomp;
    my ($r_host, $date, $uri, $http_var, $result, $size, $refara, @remainder);
    ($r_host, undef, undef, $date, undef, undef, $uri, $http_var, $result, $size,
 $refara, @remainder) = split /\s+/, $_;#ここは 1 行で書く
    $remainder = sprintf "@remainder";
    $remainder =~ s/\s+//go;

#######
# リザルトコードが 200-399 以外を除く
# "$uri ne '501'" は "warbird.fken" が "$uri" に result_code501 を返すため。
#######
    if($uri ne "501" && $result ne '"-"' && $result >= 200 && $result <= 399)
    {
if($uri !~ /\b\.(png|gif|jpg|css)\b/o)
{

########
# "$refara" に "http://www-fken.ise.osaka-sandai.ac.jp/.*"
# の文字がついている場合があるので除く。
########
    $refara =~ s/"//go;                                          #"
    $refara =~ s#http://www-fken.ise.osaka-sandai.ac.jp|
http://www.fken.ise.osaka-sandai.ac.jp##;# ここは 1 行で書く
    $date =~ s/\[//;
    my($mday, $mon, $year, $hour, $min, $sec) = split m#/|:#,$date;
    $date = "$year$mon_list{$mon}$mday";
    push(@log,{ r_host    => "$r_host",
date      => "$date",
hour      => "$hour",
min       => "$min",
sec       => "$sec",
uri       => "$uri",
```

```
http_var  => "$http_var",
refara    => "$refara",
remainder => "$remainder"
}
 );
}
    }
}
    }
    close(LOG);
    @list = (@log);
    my $line;
    foreach $line (sort Sorted @list) {
push(@sortlog,"$line->{r_host} $line->{date} $line->{hour} $line->{min} $line->{sec}
$line->{uri} $line->{http_var} $line->{refara} $line->{remainder}\n");#ここは 1 行で書く
    }

#######
# 閲覧者毎にソート及び、A - >B の形にする
#######

    my @change_log;
    {
########
# 閲覧側の同じアドレスのログのうち、現在の "$uri" と
# 次の行（一つ先）の "$refara" が 同じ場合のみ、表示する。
# "$refara" が無いもの（"_"）のものは無視する。
# 理由は "http://www-fken.ise.osaka-sandai.ac.jp"
# 以下のページを閲覧（ページ A とする）した後に、
# 他のページを閲覧し、その後に再度
# "http://www-fken.ise.osaka-sandai.ac.jp" 以下の
# ページを閲覧（ページ B とする）した場合に、A -> B となるのを
# 防ぐためである
########
my $old_uri = 'x';
my @each_host_log;
my $old_host = 'x';
my @old_each_host_log;

foreach(@sortlog)
{
    my($r_host) = split /\s+/, $_;
    if($old_host eq $r_host)
    {
```

```
push(@each_host_log, $_);
    }
    else
    {
@old_each_host_log = @each_host_log;
Makepage(@old_each_host_log);
@each_host_log = ();
push(@each_host_log, $_);
$old_host = $r_host;
    }
}
    }
    Result();
    Http();
    `rm ${FILE}link`; #アクセスパターンのファイルを消す
    `rm ${FILE}tmp`; # HTML に出力する状態のアクセスパターンのファイルを消す
}

sub Sorted
{
    ($a->{r_host} cmp $b->{r_host})
or
    ($a->{http_var} cmp $b->{http_var})
or
    ($a->{remainder} cmp $b->{remainder})
or
    ($a->{date} cmp $b->{date})
 or
    ($a->{hour} cmp $b->{hour})
 or
    ($a->{min} cmp $b->{min})
 or
    ($a->{sec} cmp $b->{sec})
}

#######
# ファイルに閲覧者のアクセスパターンを書き出すサブルーチン
# 1 時間以上経過したアクセスは無効としている
# ${FILE}link ファイルに書き出し、後でこのファイルは消している
#######
sub Makepage()
{
    my @each_host_log = @_;
    my $prev_remainder;
```

```
    my $prev_http_var = 'x';
    my $prev_access_time;
    my $read_time;
    my $prev_uri;
    my $uri_type;
    my $switch = 0;
    open(OUT, ">> ${FILE}link") or die "$!\n";
    {
foreach(@each_host_log)
{
    my($r_host, $date, $hour, $min, $sec, $uri, $http_var, $refara, $remainder)
 = split /\s+/, $_;#ここは 1 行で書く
    my $access_time = $hour * 3600 + $min * 60 + $sec;
    if($prev_http_var eq $http_var && ($prev_remainder eq $remainder
&& $access_patern[0] ne $uri))#ここは 1 行で書く
    {
$read_time = $access_time - $prev_access_time;
if($prev_uri = $refara && ($prev_uri ne $uri && ($read_time <= 3600 && $read_time > 0)))
{
    if($access_patern[$#access_patern] eq $uri)
    {
print OUT " -> $uri";
$switch = 1;
    }
    else
    {
if($switch)
        {
    $switch = 0;
    print OUT "\n#$r_host\n$uri";
}
else
        {
    print OUT " -> $uri";
}
    }
    $uri_type = "$prev_uri -> $uri";
    $prev_uri = $uri;
    $prev_access_time = $access_time;
}
elsif($prev_uri = $refara && ($read_time <= 3600 && $prev_uri eq $uri))
{
    $prev_uri = $uri;
    $prev_access_time = $access_time;
```

```
}
else
{
    print OUT "\n#$r_host\n$uri";
    $prev_uri = $uri;
    $prev_access_time = $access_time;
}
    }
    else
    {
print OUT "\n#$r_host\n$uri";
$prev_uri = $uri;
$prev_remainder = $remainder;
$prev_http_var = $http_var;
$prev_access_time = $access_time;
    }
}
    }
    close(OUT);
}

#######
# $LIST からアクセスパターンを読み込むためのサブルーチン
# ファイルには製作者が意図するアクセスパターンが書かれている
#######
sub Patern()
{
    my $prev_uri;
    open(PATERN, "< $LIST")or die $!;
    {
my $i = 1;
while(defined(my $uri = <PATERN>))
{
    chomp($uri);
    $uri =~ s#http://www-fken.ise.osaka-sandai.ac.jp##;
    push(@access_patern, $uri);
}
    }
    close(PATERN);
}

#######
# Web ページ製作者の意図するアクセスの遷移の部分を抜き取り、
# 同時に HTTP 形式のファイルに出力できる状態にする。
```

```
#######
sub Result()
{
    open(OUT, "> ${FILE}tmp") or die "$!\n";
    {
open(URL, "< ${FILE}link") or die "$!\n";
{
    my $uri;
    my $line;
    my $i = 0;
    my $n;
    my $count;
    $" = ""; #"
    while(defined($uri = <URL>))
    {
if($uri =~ m#\s+->\s+# && $uri !~ /#/)
{
    for($n = 0; $#access_patern > $n; $n++)
    {
$count = ($uri =~ s#($access_patern[$n])(\s+->\s+)($access_patern[${n}+1])(\s+)#
<b><font color="FF0000">$1</font></b>$2<b><font color="FF0000">$3</font></b>$4#g);# ここ
は 1 行で書く
$i += $count * 2;
    }
    for($n = 0; $#access_patern > $n; $n++)
    {
$count = ($uri =~ s#(<b><font color="FF0000">$access_patern[$n]</font></b>)
(\s+->\s+)($access_patern[${n}+1])(\s+)#$1$2<b><font color="FF0000">$3<\/font><\/b>$4#g);
#ここは 1 行で書く
$i += $count;
    }
chomp($uri);

#######
# Web コンテンツ製作者の意図に沿ったアクセスパターンを抜きだしている
# A は Web コンテンツ製作者の意図に沿った URL を示し、
# B はそれ以外の URL を示す。
#######
    if($i >= 1)
    {
my @line = ();
for($n = 1; $i >= $n; $n++)
{
# A -> A ...
```

```
    if($uri =~ m#^(<b><font color="FF0000">\S+</font></b>\s+->\s+)<b>#)
    {
push(@line, $1);
my $s1 = "$1";
$uri =~ s#(.*)$s1(.+)#$2#;
    }
# A $
    elsif($uri =~ m#(<b><font color="FF0000">\S+</font></b>)$#)
    {
push(@line, $1);
                                my $s1 = "$1";
$uri =~ s#(.*)$s1(.*)#$2#;
print OUT "@line\n";
@line = ();
    }
# A -> B $
    elsif($uri =~ m#(<b><font color="FF0000">\S+</font></b>)(\s+->\s+(\S+))$#)
    {
push(@line, $1, $2);
my $s1 = "$1$2";
$uri =~ s#(.*)$s1(.*)#$2#;
print OUT "@line\n";
@line = ();
    }
# A -> B ...
    elsif($uri =~ m#(<b><font color="FF0000">\S+</font></b>)(\s+->\s+(\S+).*)#)
    {
my $a = $3;
push(@line, $1, $2);
my $s1 = "$1";
$uri =~ s#(.*)$s1(.*)#$a$2#;
print OUT "@line\n";
@line = ();
    }
}
$i = 0;
    }
    else
    {
next;
    }
}
    }
}
```

```
close(URL);
    }
    close(OUT);
    $" = " "; #"
}

#######
# html ファイルを作る
#######
sub Http()
{
    my $all_patern;
    my %patern_type;
    open(URL, "< ${FILE}tmp") or die "$!\n";
    {
while(defined(my $uri = <URL>))
{
#######
# "$access_patern[0]" から始まるアクセスパターンのみを取り出し、
# その合計を $all_patern に入れる
#######
    if($uri =~ m#^<b><font color=\"FF0000\">$access_patern[0]<.+#) #"
    {
chomp($uri);
$patern_type{$uri}++;
$all_patern++;
    }
}
    }
    close(URL);

    my $line;
    my @line;
    foreach $line (@access_patern)
    {
push(@line, "<b><font color=\"FF0000\">$line</font></b>")
    }
    my $main_patern = "@line";
    $main_patern =~ s#0\.000000E\+00#%7E#go; # "%7E" を元に戻すため
    $main_patern =~ s#</b>(\s)<b>#</b>$1->$1<b>#go; # "s/ / -> /"

    open(HTTP2, "> ${KEKKA}kekka.html") or die "$!\n";
    {
print HTTP2 qq(<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">\n);
```

```
print HTTP2 qq(<HTML lang="ja">\n);
print HTTP2 qq(<HEAD>\n);
print HTTP2 qq(<META http-equiv="Content-Type" content="text/html; charset=EUC">\n);
print HTTP2 qq(<TITLE>解析結果<\/TITLE>\n);
print HTTP2 qq(<\/HEAD>\n);
print HTTP2 qq(<BODY>\n);
print HTTP2 qq(<h2>解析結果<\/h2>\n);

print HTTP2 qq(<hr>\n);
print HTTP2 qq(製作者の意図するアクセスパターン<br>\n);
if(!$patern_type{$main_patern})
{
    $patern_type{$main_patern} = 0;
}

#######
# $main_patern が 0 の場合には除算が出来ないので 0 %を入れておく
#######
my $percent;
if($patern_type{$main_patern} == 0)
{
   $percent = 0;
}
else
{
    $percent = int($patern_type{$main_patern} / $all_patern * 100);
}
print HTTP2 qq($main_patern :$patern_type{$main_patern}回 :$percent %<br>\n);
print HTTP2 qq(<hr>\n);

print HTTP2 qq(途中までは製作者の意図通りにアクセスしていた閲覧者のアクセスパターン<br>\n);
my $key;
my $patern;
for(my $i = 0; $#access_patern > $i; $i++)
{
    $patern = "$line[$i] -> (?!<b>)";
    foreach $key(keys %patern_type)
    {
if($key ne $main_patern && $key =~ $patern)
{
    if($key =~ m#($line[$i] -> \S+)#)
    {
$percent = int($patern_type{$key} / $all_patern * 100);
print HTTP2 qq($1 :$patern_type{$key}回 :$percent %<br>\n);
```

```
     }
}
    }
}
print HTTP2 qq(<hr>\n);
print HTTP2 qq(<\/BODY>\n);
print HTTP2 qq(<\/HTML>\n);
    }
    close(HTTP2);
}
```

————ここまで————