

Techniques de Programmation pour Internet

Exemples sur Perl

DESS Gnie Informatique
Universit Joseph Fourier

1997 – 1998

Jrme Martin

1 Recherche d'un mot cl dans les fichiers texte d'une arborescence.

1.1 Parcours de l'arborescence.

Il s'agit de parcourir une arborescence disque en affichant les noms des fichiers/repertoires. La fonction `search` peut tre crite :

```
sub search ($)
{
    local ($path) = $_[0];

    if (!(~l $path))
    # Elimination des liens
    {
        if (-d $path)
        {
            opendir THISDIR, $path or
                die "Erreur pour ouvrir le repertoire $path\n";
            local (@allfiles) = readdir THISDIR;
            close THISDIR;

            foreach $file (@allfiles)
            {
                print "$path/$file\n";
                search ("$path/$file")
                    if (($file ne ".") and ($file ne ".."));
            }
        }
    }
}
```

Un appel possible de cette fonction est :

```
search($ENV{PWD})
```

1.2 Affichage des fichiers texte par ordre alphabtique

Dans un premier temps, une liste vide destine empiler les fichiers texte est dfinie :

```
local (@textfiles) = ();
```

La fonction `search` est alors modifie :

```
sub search ($)
{
    local ($path) = $_[0];

    if (!(~l $path))
    # Elimination des liens
    {
        if (-d $path)
        {
            opendir THISDIR, $path or
```

```

    die "Erreur pour ouvrir le repertoire $path\n";
    local (@allfiles) = readdir THISDIR;
    close THISDIR;

    foreach $file (@allfiles)
    {
        search ("$path/$file")
            if (($file ne ".") and ($file ne ".."));
    }
}
elseif ((-f $path) and (-T $path))
# plain file & text
{
    push @textfiles, $path;
}
}
}

```

L'appel de la fonction et l'affichage du resultat sont raliss par :

```

sub Display (@)
{
    foreach $f (sort @_ )
    {
        print "$f\n";
    }
}

```

```

Display (@textfiles);
search ($ENV{PWD});

```

1.2.1 Affichage tri par suffixe puis alphabétique

Dans cette version, la fonction Display devient :

```

sub Display (@)
{
    foreach $f (sort sortedsuffapla @_ )
    {
        print "$f\n";
    }
}

```

o sortedsuffapla est la fonction de tri dfinie par :

```

sub sortedsuffapla
{
    local ($namea, $suffa, $nameb, $suffb);
    if ($a =~ /\.*\/(.*)\.(.*)/)
    {
        $namea = $1;
        $suffa = $2;
    }
    if ($b =~ /\.*\/(.*)\.(.*)/)

```

```

    {
        $nameb = $1;
        $suffb = $2;
    }

    return ($suffa cmp $suffb) and ($namea cmp $nameb);
}

```

1.3 Affichage du nom du fichier et son type.

Nous disposons d'une liste associative %filetype dfinie par exemple par

```

local (%filetype) = ("pl", "Fichier Perl",
                    "c", "Fichier C",
                    "cc", "Fichier C++",
                    "cpp", "Fichier C++",
                    "pas", "Fichier Pascal",
                    "h", "Fichier entete C",
                    "ps", "Fichier Postscript",
                    "html", "Fichier HTML");

```

L'affichage du rsultat par la fonction Display peut tre modifi en :

```

sub Display (@)
{
    foreach $f (sort @_)
    {
        PrintFile $f;
    }
}

```

o la fonction PrintFile est dfinie par :

```

sub PrintFile ($)
{
    local ($file) = $_[0];
    local ($type);

    $file =~ s/.*\///;
    $type = $file;
    $type =~ s/.*\.(.*)/if (exists $filetype{$1})
        {
            $filetype{$1}
        } else {
            "Inconnu"
        } /e;

    print "$file -> $type\n";
}

```

1.4 Recherche d'une expression rgulire dans un fichier texte

On cherche une expression rgulire stocke dans la variable \$exp sur l'ensemble des fichiers texte de l'arborescence. Dans le programme prcdent, la ligne push @textfiles, \$path; est remplacpar ParseFile \$path; La fonction ParseFile est dfinie par :

```
sub ParseFile ($)
{
    local ($file) = $_[0];

    open (FILEIN, $file) or die "Ouverture du fichier impossible\n";
    while (<FILEIN>)
    {
        push @selectedfiles, $path if (/$exp/);
    }
    close (FILEIN);
}
```

La variable peut, par exemple, être définie par :

```
local ($exp) = "foreach\\s\\\$\\w{2,}";
```

2 Calcul des statistiques d'accs au serveur.

Nous voulons dnombtrer les accs raliss sur notre serveur Web. Pour cela, nous souhaitons comptabiliser le nombre d'accs par domaine, par page et une liste de tous les accs raliss et tris par domaine, machine et date. Ces statistiques devront tre ralises partir d'une date spcifie dans le programme. Elles sont calcules partir des informations du fichier `logs/access_logs` qui contient l'ensemble des accs raliss sur le serveur. Chaque ligne de ce fichier est de la forme :

```
pandora.imag.fr- [13/Dec/1997:18:30:11+0100] "GET/~jmartin/DESS-GI/HTTP/1.0" 200 601
```

Nous devons donc :

1. Lire un fichier contenant la description des domaines (association suffixes/nom),
2. Calculer le domaine partir d'un nom de machine,
3. Lire le fichier des accs,
4. Gnrer les rsultats dans un format HTML.

2.1 Lecture du fichier de description des domaines.

Les domaines sont dfinis dans le fichier dont le nom est contenu dans la variable `$domainDesc`. Chaque ligne a la syntaxe : "*suffixe_domaine\t nom_domaine*".

```
Exemple : | AD    Andore
           | AE    Etats Arabes Unis
           | AF    Afghanistan
           | ...
```

La fonction `ReadDomainName` effectue cette lecture en stockant le rsultat dans la liste associative `%domainName` qui tablit la relation suffixe—nom.

```
sub ReadDomainsName
{
  open (FILEIN, "$domainDesc") ||
    die "Erreur d'ouverture du fichier $domainDesc\n";
  while (<FILEIN>)
  {
    $_ =~ s/(\w*)\s*(\w*)/$1\+$2/;
    local ($abbr, $name) = split (/\/+);
    $abbr =~ tr/[A-Z]/[a-z]/;
    chop $name;
    $domainName{$abbr} = $name;
  }
  $domainName{'adresse IP'} = "Adresse IP";
  close (FILEIN);
}
```

2.2 Calcul du domaine partir d'un nom de machine.

Une machine est identifie par une adresse IP ou par son nom symbolique. De son adresse IP, il n'est pas possible de trouver son domaine. Pour les noms symboliques, le domaine est donn par la dernire partie de son nom.

La fonction pour rcuprer le domaine partir d'un nom de machine est donne par la fonction `GetDomain` :

```

sub GetDomain
{
    local ($domain) = $_[0];

    return "Adresse IP" if ($domain =~ /^[0-9]+.*\/);

    $domain =~ s/.*\./g;
    $domain =~ tr/[A-Z]/a-z/;
    return $domain;
}

```

2.3 Lecture du fichier des accs.

La lecture du fichier des accs consiste lire le fichier texte, vérifier les origines des accs (nous voulons exclure tous les accs des machines de notre site) et les fichiers accds (exclusion des fichiers images) puis comptabiliser les statistiques. Ces statistiques sont stockées, par exemple, dans les variables et listes :

```

$accessCount  Nombre d'accs totaux,
@accesses     Listes des accs correspondant aux critères donnés,
%domains      Nombre d'accs depuis le domaine donné,
%pages        Nombre d'accs la page donnée,
%hosts        Nombre d'accs depuis le site donné (pour générer la liste de tous les accs),

```

```

sub ReadLogFile
{
    open (FILEIN, "<$accessLogFile") ||
        die "Erreur à l'ouverture du fichier '$accessLogFile'\n";

    while (<FILEIN>)
    {
        if (&AccessVerified)
        {
            local ($remote, $dash1, $dash2, $date, $gmt,
                $type, $page) = split / /;
            # La date a le format JJ/MMM/AAAA:HH:MM:SS
            if (DateVerified ($dateStart, $date))
            {
                local ($domain) = GetDomain ($remote);

                $accessCount++;
                $pages{$page}++;
                $hosts{$remote}++;
                $domains{$domain}++;
                push @accesses,
                    "$domainName{$domain}+$remote+$date+$page";
            }
        }
    }
    close (FILEIN);
}

```

2.3.1 Fonction de vrification des accs.

Cette fonction doit vrifier que la machine accdant au serveur est comptabilise. Par exemple par la chane de caractres `$excludeHosts` contenant une expression rgulire dfinissant les machines non compatbilises.

```
Exemple : | $excludeHosts = "boole|babbage|frege";
```

De la mme manire, les fichiers comptabiliss et non comptabiliss peuvent tre dfinis par les variables `$excludeURL` et `$includeURL`.

```
$excludeURL = ".gif|.jpg|cgi-bin";  
$includeURL = ".html|.ps.gz|moi.jpg";
```

La fonction `AccessVerified` de vrification s'crit alors :

```
sub AccessVerified  
{  
    return (/ $includeURL/i && ( ! / $excludeURL/i ) && ( ! / $excludeHosts/i ));  
}
```

2.3.2 Fonction de vrification de date.

Cette fonction vrifie que la date donne en argument est postrieure la date de rfrence du programme et donne par la variable `$dateStart`. La fonction `DateVerified` qui vrifie la postriorit des dates est alors :

```
sub DateVerified  
{  
    local ($date0) = split (/:/, $_[0]);  
    local ($date1) = split (/:/, $_[1]);  
    local ($day0, $mounth0, $year0) = split "-", $date0;  
    local ($day1, $mounth1, $year1) = split "-", $date1;  
  
    return (($year0 < $year1) ||  
            (($year0 == $year1) &&  
             ($monthNum{$mounth0} < $monthNum{$mounth1})) ||  
            (($year0 == $year1) &&  
             ($monthNum{$mounth0} == $monthNum{$mounth1}) &&  
             ($day0 < $day1));  
}
```

Dans ce cas, la liste associative `%monthNum` permet une conversion des abbrviations de noms de mois vers un numro et est dfinie par :

```
local(%monthNum) = ('Jan', 1, 'Feb', 2, 'Mar', 3,  
                   'Apr', 4, 'May', 5,  
                   'Jun', 6, 'Jul', 7, 'Aug', 8,  
                   'Sep', 9, 'Oct', 10,  
                   'Nov', 11, 'Dec', 12);
```

2.4 Gnration des rsultats.

2.4.1 Accs par domaine.

L'affichage du rsultat des accs par domaine devant tre tri, il faut crire une procudre de tri :

```

sub domainsbyaccess
{
    $domains{$b} <=> $domains{$a};
}

```

Cette fonction est une fonction de comparaison entre les domaines \$a et \$b, elle peut ensuite tre utilise avec la fonction sort :

```

local (@domainsSorted) = sort domainsbyaccess (keys %domains);

```

Un affichage possible du rsultat est alors

```

sub DisplayAccessesByDomain
{
    foreach $dom (@domainsSorted)
    {
        local ($percen) = $domains{$dom}/$accessCount*100
        if ($accessCount > 0);
        print FILEOUT "<TR><TD>$domainName{$dom}"
        print FILEOUT "<TD ALIGN=RIGHT>$domains{$dom}"
        print FILEOUT "<TD ALIGN=RIGHT>";
        printf FILEOUT ("%3.2f", $percen);
        print FILEOUT "</TR>\n";
    }
}

```

2.4.2 Accs par pages.

Le programme permettant d'afficher l'ensemble des pages accdes en sparant les pages HTML des autres est donn par :

```

sub pagesbyaccess
{
    $pages{$b} <=> $pages{$a};
}

sub DisplayAccessesByPages
{
    foreach $p (keys %pages)
    {
        if ($p =~ /\.html$/)
        {
            push @HTMLPages, $p;
        } else {
            push @otherPages, $p;
        }
    }

    local (@HTMLSorted) = sort pagesbyaccess @HTMLPages;
    foreach $p (@HTMLSorted)
    {
        local ($percen) = $pages{$p}/$accessCount*100
        if ($accessCount > 0);
    }
}

```

```

    print FILEOUT "<TR><TD>$p"
    print FILEOUT "<TD ALIGN=RIGHT>$pages{$p}"
    print FILEOUT "<TD ALIGN=RIGHT>";
    printf FILEOUT ("%3.2f", $percen);
    print FILEOUT "</TR>\n";
}

local (@otherSorted) = sort pagesbyaccess @otherPages;
foreach $p (@otherSorted)
{
    local ($percen) = $pages{$p}/$accessCount*100
    if ($accessCount > 0);
    print FILEOUT "<TR><TD>$p"
    print FILEOUT "<TD ALIGN=RIGHT>$pages{$p}"
    print FILEOUT "<TD ALIGN=RIGHT>";
    printf FILEOUT ("%3.2f", $percen);
    print FILEOUT "</TR>\n";
}
}

```