



SAO₅ PROGRAMMATION

SAE

ARVIN-BÉROD Maxence - GIRAULT Adrien | Statistiques de Connexions | 26/10/2021

1. Analyse de acces.log :

Le fichier access.log sert à la journalisation

de tous les accès au serveur, c'est-à-dire toutes les requêtes HTTP reçues et la façon

dont elles ont été traitées. Le format de ce fichier est paramétrable via l'option

access_log du fichier squid3.conf.

Pour cette requête par exemple :

192.168.56.101 - - [10/Feb/2019:19:26:26 +0100] "GET /cgi-bin/versiondynamique-txt HTTP/1.1" 200 441 "http://192.168.56.101/" "Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0"

192.168.56.101 : adresse IP client

[10/Feb/2019:19:26:26 +0100] : Temps au format unix

GET /cgi-bin/versiondynamique-txt HTTP/1.1 : La méthode utilisé pour récupérer la ressource, et l'url de celle-ci

200 : Code message "OK" http qui signifie que la requête a été traité avec succès

441 : La taille de la donnée

http://192.168.56.101/ : l'adresse ip serveur (étant donné que je créer le serveur sur ma machine, il a mon adresse IP

"Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0" : sous quel navigateur de recherche

2. Nos étapes :

A) Le Commencement:

SAE PROGRAMMATION :

-

- On a besoin de acces.log dynamique qui sont les logs dynamiques de notre site apache 2 en 127.0.0.1

répertoire : /var/log/apache2

- On a besoin de acces.log statique dans le dossier SAE15/src

- On a besoin de index.html pour importer nos graphiques dynamiques

répertoire : /var/www/html

-On a besoin de aecrire.html qui va créer les graphes

répertoire : /usr/lib/cgi-bin

-On a besoin de doc.pdf pour les créations de graphique

répertoire : SAE15/lib

1. Nous devons créer un programme en C qui se base sur le acces.log qui comprend : le

nombre total de requêtes enregistrées depuis la date de la première connexion

(chaque ligne correspond à une requête et une répartition mensuelle)

Ensuite il doit être en :

- **WEB (par défaut)** : le programme se comporte comme un cgi, est dans le usr/lib/cgi-bin

Il génère du code html

Option graphic : il ouvre une fenêtre graphique et affiche les informations.

Option text : Le programme affiche sous forme de texte

B) Les étapes de la programmation :

Nous commençons à mettre notre dossier SAE15 dans le bureau.

Commençons à aller dans le dossier « src » et à explorer les codes en c.

La version statique peut nous aider, les fonctions qui appellent à ouvrir un fichier en c nous permettra de récupérer le fichier acces.log **dynamique** qui est dans le dossier ***/var/log/apache2/acces.log*** .

Je commence à me définir ce que je dois faire :

Algorithme programmation dynamique version textuel :

Compter le nombre de connexions sur le site:



Répartition des connexions de tout les mois depuis le début du site :



Algorithme programmation dynamique version graphique :

Compter le nombre de connexions sur le site:



Répartition des connexions de tout les mois depuis le début du site : ✓

Algorithme programmation dynamique version graphique web:

Compter le nombre de connexions sur le site: ✓

Répartition des connexions de tout les mois depuis le début du site : ✓

2) A) Partie dynamique version textuel:

1) Avec toutes les informations que nous avons récoltés via le sujet en pdf, plus, les informations que nous pouvons récupéré sur internet nous commençons le pseudo langage de notre programme (ce sera un peu plus bas)

Puis, voici le début de notre programme, la partie 1, le comptage du nombre de connexions de acces.log :

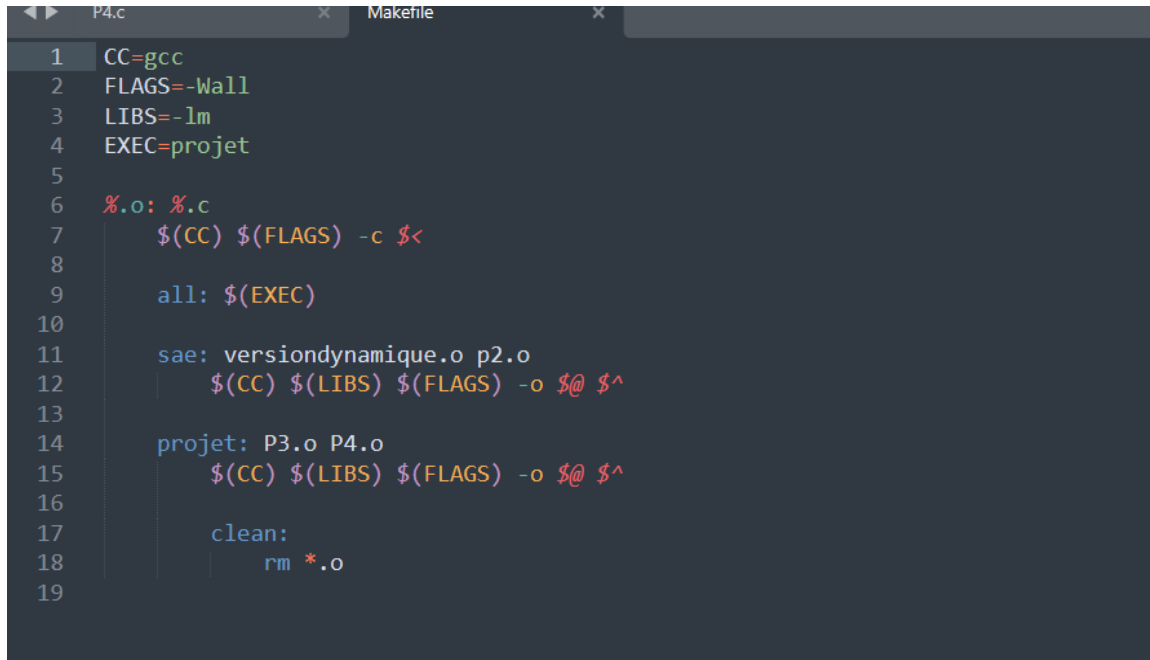
```
1 //include <stdio.h> //indique au préprocesseur d'inclure le contenu de la bibliothèque stdio.h
2 #include <stdlib.h> //déclare des fonctions qui effectuent la conversion de nombres, la gestion de la mémoire et d'autres tâches.
3 #include <string.h> /* contient les définitions des macros, des constantes et les déclarations de fonctions et de types de chaînes de caractères,
4 mais aussi pour diverses fonctions de manipulations de la mémoire.*/
5
6 int main() { // déclaration de ma fonction
7     int c; // création de ma variable entière "c"
8     int nb_line = 0; // création de ma variable entière "nb_line"
9     FILE * file = fopen("/var/log/apache2/access.log", "r"); /* ouverture de mon fichier que je désire en mode "r", lecture seule */
10    /* Début de la boucle "tant que" */
11    while ((c = getc(file)) != EOF) { /* tant que ma variable c lis mon fichier, et que ce n'est pas égal
12    à EOF, ça lis le fichier */
13        if (c == '\n') /* Incrémente le "nb_line" si ce caractère est une nouvelle ligne.*/
14            ++nb_line; /* incrémente 1 à "nb_line" */
15    }
16
17    printf("Depuis le 21/Feb/2018 on a enregistré %d connexions.\n", nb_line); /* Afficher le nombres de connexions totales depuis le début */
18    return 0;
19 }
20
21
```

```
user@debian: ~/SAE/SAE15/src$ gcc -m-version-script=/etc/ld.so.conf.d/x86_64-linux-gnu.conf -fPIC -fPIE -pie -Wl,-rpath=/usr/local/lib -o a.out a.c
user@debian: ~/SAE/SAE15/src$ ./a.out
Depuis le 21/Feb/2018 on a enregistré 21917 connexions.
user@debian: ~/SAE/SAE15/src$
```

```
1 #Déclarons nos variables :
2 CC=gcc #Ma variable de compilateur pour le C
3 FLAGS=-Wall #Regrouper les options de compilation en C
4 LIBS=-lm #utile si vous utiliser <math.h>
5 EXEC=sae #Contient le nom des exécutables à gérer
6 #-----
7 %.o: %.c #constitue un .o à partir d'un .c
8     $(CC) $(FLAGS) -c $< #utilisons des variables et la variable interne $< = le nom de la première dépendance
9
10
11 all: $(EXEC)
12
13 sae: P1.o
14     $(CC) $(LIBS) $(FLAGS) -o $@ $^ #variable $@ = le nom de la cible et $^ = la liste des dépendances
15
16
17
18
19 clean:
20     rm *.o
21
22
```

C'est déjà un début, nous voyons que nous avons pas de makefile, nous en faisons un de suite, V1 :

Voici le Makefile re modifié pour notre projet, V2 (il changera au fur et à mesure que nous avançons dans le projet) :



```
1 CC=gcc
2 FLAGS=-Wall
3 LIBS=-lm
4 EXEC=projet
5
6 %.o: %.c
7     $(CC) $(FLAGS) -c $<
8
9 all: $(EXEC)
10
11 sae: versiondynamique.o p2.o
12     $(CC) $(LIBS) $(FLAGS) -o $@ $^
13
14 projet: P3.o P4.o
15     $(CC) $(LIBS) $(FLAGS) -o $@ $^
16
17 clean:
18     rm *.o
19
```

Voici le pseudo langage de ce code :

VARIABLES

c ← réel et nblne ← réel

DEBUT

Ouvrir fichier access.log et le lire

Tant que « c » lis mon fichier et n'est pas égal à EOF (end of file)

si « c » reviens à la ligne

alors il faut incrémenter de 1 la variable « nblne »

Afficher "Depuis le 21/Feb/2018 on a enregistré x connexions."

FIN

2) Désormais, commençons la suite, il faudra Répartir les connexions de tout les mois depuis le début du site :

Faisons notre pseudo langage :

VARIABLES

déclaration de toutes mes variables de mois en réel

déclaration de toutes mes variables de mois pour les % en float

c ← en réel

log ← en char

tab[29] ← en char comportant 29 caractères (car cela va jusqu'à l'année dans la ligne)

separateur ← en réel

nbligne ← en réel

nbligne1 ← en float

DEBUT

Ouvrir fichier access.log et le lire

tant que le fichier est ouvert

mettre « separateur » à 0

on créer un variable interne i en réel

pour i = 0, i inférieur à 29, i=i+1

tab i ← log

lire dans le fichier les caractères de log

tant que « log » ne reviens pas à ligne

lire dans le fichier les caractères de log

pointeur p pour décomposer la chaîne du tableau aux « / » entre les mois

tant que « p » n'est pas égal à NULL

si le separateur est égal à 1

si on compare le pointeur p et le texte « Jan » et que c'est égal à 0

alors jan=jan+1

si on compare le pointeur p et le texte «Fev» et que c'est égal à 0

alors fev=fev+1

etc on continue avec tout les mois ...

le pointeur est égal à la séparation pointant sur NULL et le texte « / »

separateur = separateur +1

Les pourcentages :

on initialise nbline égal à tous les mois de l'année

nbline1 sera égal à nbline

*puis, la variable « jan_100 » sera égal à « formule du pourcentage » =
(jan/nbline1)*100*

et ainsi de suite pour chaque mois ...

Compter les lignes du fichier :

Tant que « c » lis mon fichier et n'est pas égal à EOF (end of file)

si « c » reviens à la ligne

alors il faut incrémenter de 1 la variable « nbligne »

Afficher "Depuis le 21/Feb/2018 on a enregistré x connexions."

afficher « Jan : % »

etc pour tout les mois (cela va permettre d'afficher en pourcentage le nombre de connexions par mois)

FIN

Voici le programme :

```
mais aussi pour diverses fonctions de manipulations de la mémoire.*/
#include "libgraphique.h"
#include <math.h>

void versiontexte() { // déclaration de ma fonction

    //déclaration de mes vaiaables de mois
    int jan=0, feb=0, mar=0, apr=0, may=0, jun=0, jul=0, aug=0, sep=0, oct=0, nov=0, dec=0;
    float jan_100=0, feb_100=0, mar_100=0, apr_100=0, may_100=0, jun_100=0, jul_100=0, aug_100=0, sep_100=0, oct_100=0, nov_100=0, dec_100=0;

    int c; // création de ma variable entière "c"
    char log; /* Création de ma variable log pour compter caractère par caractère*/
    char tab[29]; /* Création de mon tableau de 29 caractères (jusqu'à l'année dans le accesslog) */
    int separateur=0; /* Séparateur des premières chaînes de caractères */
    int nbligne = 0; // création de ma variable entière "nbligne" pour compter le nombre de connexions
    float nbligne1; // variable permettant de regrouper tous les mois ensemble

    FILE * file = fopen("/var/log/apache2/access_log" "r"); /* ouverture de mon fichier que je désire en mode "r" lecture seule */
    while (p != NULL) { // tant que le pointeur ne pointe pas sur rien c'est vrai
        if (separateur == 1) {
            if (strcmp(p, "Jan") == 0)
                jan = jan + 1;

            if (strcmp(p, "Feb") == 0)
                feb = feb + 1;

            if (strcmp(p, "Mar") == 0)
                mar = mar + 1;

            if (strcmp(p, "Apr") == 0)
                apr = apr + 1;

            if (strcmp(p, "May") == 0)
                may = may + 1;

            if (strcmp(p, "Jun") == 0)
                jun = jun + 1;

            if (strcmp(p, "Jul") == 0)
                jul = jul + 1;

            if (strcmp(p, "Aug") == 0)
                aug = aug + 1;

            if (strcmp(p, "Sep") == 0)
                sep = sep + 1;

            if (strcmp(p, "Oct") == 0)
                oct = oct + 1;

            if (strcmp(p, "Nov") == 0)
                nov = nov + 1;
```

```

        dec = dec + 1;
    }

    p = strtok(NULL, "/");
    separateur = separateur + 1;
}
// Les pourcentages :
nblne = jan+feb+mar+apr+may+jun+jul+aug+sep+oct+nov+dec;
nblne1 = nblne;
jan_100=(jan/nblne1)*100;
feb_100=(feb/nblne1)*100;
mar_100=(mar/nblne1)*100;
apr_100=(apr/nblne1)*100;
may_100=(may/nblne1)*100;
jun_100=(jun/nblne1)*100;
jul_100=(jul/nblne1)*100;
aug_100=(aug/nblne1)*100;
sep_100=(sep/nblne1)*100;
oct_100=(oct/nblne1)*100;
nov_100=(nov/nblne1)*100;
dec_100=(dec/nblne1)*100;
}

/* Début de la boucle "tant que" permettant de compter les lignes du fichier access_log */
while ((c = getc(file)) != EOF){ /* tant que ma variable c lis mon fichier, et que ce n'est pas égal
à EOF, ça lis le fichier */
    if (c == '\n')/* Incrémente le "nblne" si ce caractère est une nouvelle ligne.*/
        nblne = nblne + 1 ;/* incrémenter 1 à "nblne" */
}

printf("Depuis le 21/Feb/2018 on a enregistré %d connexions.\n", nblne); /* Afficher le nombres de connexions totales depuis le début */

printf("Jan: %.1f%\n", jan_100);
printf("Feb: %.1f%\n", feb_100);
printf("Mar: %.1f%\n", mar_100);
printf("Apr: %.1f%\n", apr_100);
printf("May: %.1f%\n", may_100);
printf("Jun: %.1f%\n", jun_100);
printf("Jul: %.1f%\n", jul_100);
printf("Aug: %.1f%\n", aug_100);
printf("Sep: %.1f%\n", sep_100);
printf("Oct: %.1f%\n", oct_100);
printf("Nov: %.1f%\n", nov_100);
printf("Dec: %.1f%\n", dec_100);

}

/* On va avoir une boucle while car tant que y'a des lignes dans le dossier on défini une variable
j = 0 et dans la boucle for on initialise une variable i = 0 et tant que i<= */

```

D'autres informations en gris seront ajoutés directement dans le code afin de comprendre ce qu'on fait ligne par ligne

Voici la partie test :

```
user@debian:~/SAE/SAE15/STC$ ./P4
Depuis le 21/Feb/2018 on a enregistré 21917 connexions.
Jan: 10.7%
Feb: 18.1%
Mar: 6.5%
Apr: 5.4%
May: 5.4%
Jun: 3.4%
Jul: 2.0%
Aug: 0.4%
Sep: 16.3%
Oct: 11.7%
Nov: 6.6%
Dec: 13.4%
```

Nous avons bien **réaliser** la **première** étape qui est la version **TXT**. Les explications du code sont présentes dans le code (sur le moodle)

Pour notre partie de code nous devons :

1- Extraire tout les mois de chaque ligne depuis le début du fichier access.log 

2- Compter le nombre de fois ou la chaîne de caractère de chaque mois apparaît ✓

3- Convertir ces chiffres en % ✓

3) A) Partie dynamique version graphique :

La partie code ajouté à notre programme :

```
445 Point p = {30,350}; // Création d'un point nommé p de champ x = 30 et champ y = 350
446 ouvrir_fenetre(430,350); /* Création de la fenetre de 430 de largeur et 350 de hauteur */
447 dessiner_rectangle((Point){0,0},430,350,blanc); // création d'un rectangle de couleur blanche ou le point Point se trouve en haut à gauche et couvre
448 // Chaînes de caractères
449 char Pcompteur[50], Pjan[10], Pfev[10], Pmar[10], Pavr[10], Pmai[10], Pjun[10], Pjul[10], Paou[10], Psep[10], Poct[10], Pnov[10], Pdec[10];
450 sprintf(Pcompteur,"Depuis le 01 Jan 2018 on a %d connexions.", nbline); // Afficher sur l'interface le nombre de connexion dynamiquement
451
452 // Permettre d'arrondir un float en le transformant en entier int
453 jane = floor(jan_100);
454 feve = floor(feb_100);
455 mare = floor(mar_100);
456 apre = floor(apr_100);
457 maye = floor(may_100);
458 june = floor(jun_100);
459 jule = floor(jul_100);
460 auge = floor(aug_100);
461 sepe = floor(sep_100);
462 octe = floor(oct_100);
463 nove = floor(nov_100);
464 dece = floor(dec_100);
465 // Permettre de stocker dans les chaînes de caractères Pjan,etc
466 sprintf(Pjan, "%d", jane);
467 sprintf(Pfev, "%d", feve);
468 sprintf(Pmar, "%d", mare);
469 sprintf(Pavr, "%d", apre);
470 sprintf(Pmai, "%d", maye);
471 sprintf(Pjun, "%d", june);
472 sprintf(Pjul, "%d", jule);
473 sprintf(Paou, "%d", auge);
474 sprintf(Psep, "%d", sepe);
475 sprintf(Poct, "%d", octe);
476 sprintf(Pnov, "%d", nove);
477 sprintf(Pdec, "%d", dece);
478 afficher_texte(Pcompteur,12,p,noir);
479
```

```
481
482 p.y=300;
483 // Afficher les mois en dessous des graphes
484 afficher_texte(" Jan Fev Mar Avr Mai Juin Jul Aou Sep Oct Nov Dec",14,p,noir);
485 // Dessiner tout les graphes en fonction du nombre de pourcentage
486 dessiner_rectangle((Point){30,(300-jan_100*10)},30,(jan_100*10),bleu);
487 dessiner_rectangle((Point){61,(300-feb_100*10)},30,(feb_100*10),bleu);
488 dessiner_rectangle((Point){92,(300-mar_100*10)},30,(mar_100*10),bleu);
489 dessiner_rectangle((Point){123,(300-apr_100*10)},30,(apr_100*10),bleu);
490 dessiner_rectangle((Point){154,(300-may_100*10)},30,(may_100*10),bleu);
491 dessiner_rectangle((Point){185,(300-jun_100*10)},30,(jun_100*10),bleu);
492 dessiner_rectangle((Point){216,(300-jul_100*10)},30,(jul_100*10),bleu);
493 dessiner_rectangle((Point){247,(300-aug_100*10)},30,(aug_100*10),bleu);
494 dessiner_rectangle((Point){278,(300-sep_100*10)},30,(sep_100*10),bleu);
495 dessiner_rectangle((Point){309,(300-oct_100*10)},30,(oct_100*10),bleu);
496 dessiner_rectangle((Point){340,(300-nov_100*10)},30,(nov_100*10),bleu);
497 dessiner_rectangle((Point){371,(300-dec_100*10)},30,(dec_100*10),bleu);
498 // Afficher les pourcentages en haut des graphes
499 afficher_texte(Pjan,14,(Point){40,100},noir);
500 afficher_texte(Pfev,14,(Point){71,100},noir);
501 afficher_texte(Pmar,14,(Point){97,100},noir);
502 afficher_texte(Pavr,14,(Point){127,100},noir);
503 afficher_texte(Pmai,14,(Point){159,100},noir);
504 afficher_texte(Pjun,14,(Point){189,100},noir);
505 afficher_texte(Pjul,14,(Point){220,100},noir);
506 afficher_texte(Paou,14,(Point){252,100},noir);
507 afficher_texte(Psep,14,(Point){282,100},noir);
508 afficher_texte(Poct,14,(Point){313,100},noir);
509 afficher_texte(Pnov,14,(Point){345,100},noir);
510 afficher_texte(Pdec,14,(Point){385,100},noir);
511
512 actualiser(); // actualiser
513 attendre_clic();
514 fermer_fenetre(); // fermer la fenetre si la personne clique
515 }
```

Explication du code (pseudo code qui bous permet de savoir ce qu'on doit faire étape par étape):

Création d'un point nommé p de champ x = 30 et champ y = 50

Création de la fenetre de 430 de largeur et 350 de hauteur

création d'un rectangle de couleur blanche ou le point Point se trouve en haut à gauche et couvre toute l'interface graphique

Créer un tableau Pcompteur de 50 caractères et etc pour les autres tableaux

Afficher sur l'interface le nombre de connexion dynamiquement

Permettre d'arrondir un float en le transformant en entier int

Permettre de stocker dans les chaînes de caractères Pjan,etc

Afficher les mois en dessous des graphes

Dessiner tout les graphes en fonction du nombre de pourcentage

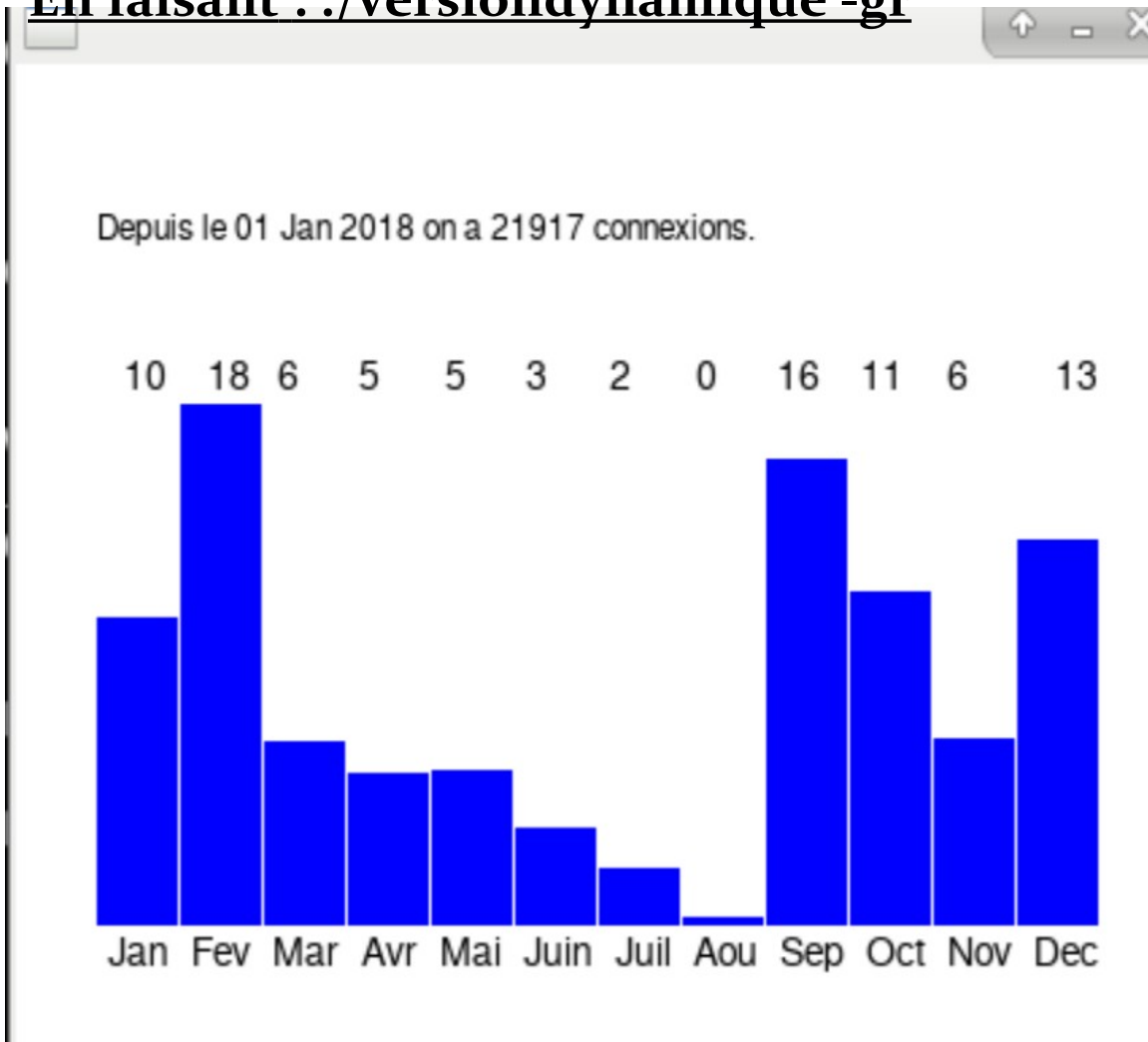
Actualiser

Fermer la fenetre si la personne clique

Nous avons donc crée la fonction void versiongraphique pour que cela marche

Voici la partie exécution :

En faisant : ./versiondynamique -gr



4) A)Partie dynamique version WEB :

Pour le HTML :

- Récupérer le code statique html

- Fonction lecture style et version web dans le code statique
- Transforme % en Pixel
- Dossier /usr/lib/cgi-bin (.html et executable (nom versiondynamique) du prof)

Nous allons continuer d'ajouter à notre programme une fonction . Celle-ci s'appellera : versionweb()

Nous avons repris du code de la versionstatique pour s'aider. C'est aller plus vite. Sans ça, cela aurait pris un peu plus de temps.

Nous avons rajouter 10* pour multiplier par 10 le résultat des pourcentages pour que les pixels soient proportionnels

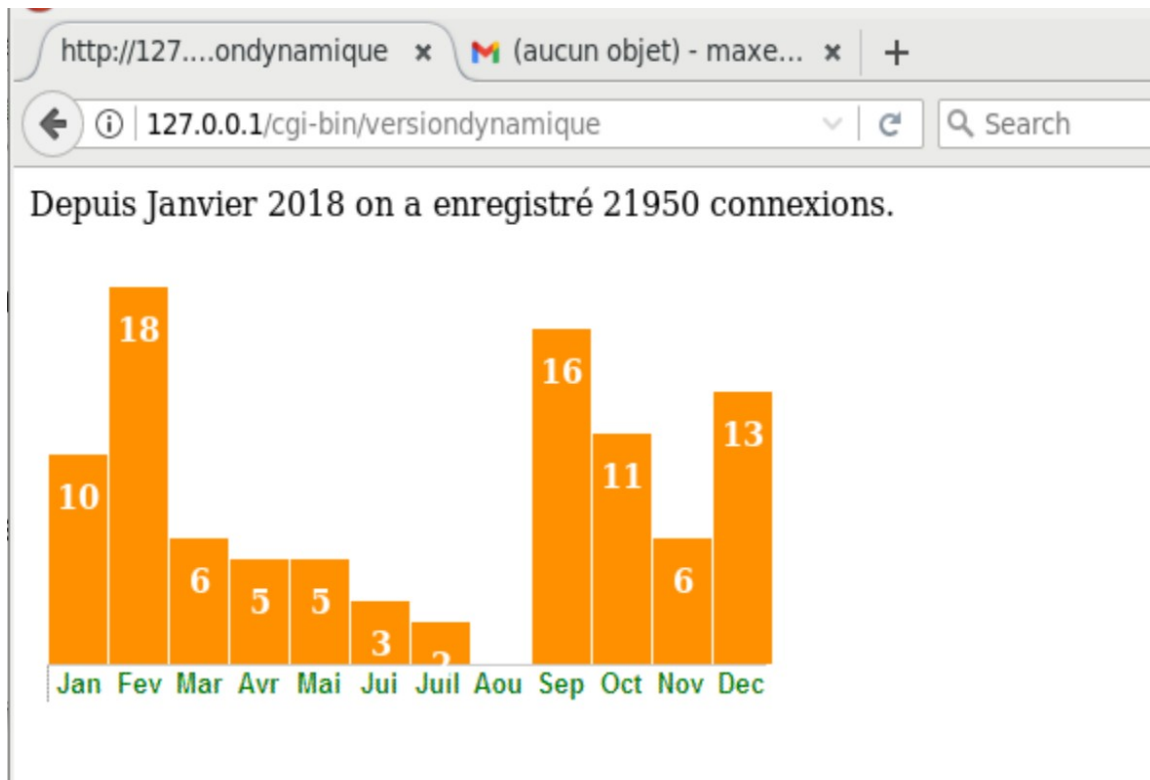
```
}  
  
jane = 10*floor(jan_100);  
feve = 10*floor(feb_100);  
mare = 10*floor(mar_100);  
apre = 10*floor(apr_100);  
maye = 10*floor(may_100);  
june = 10*floor(jun_100);  
jule = 10*floor(jul_100);  
auge = 10*floor(aug_100);  
sepe = 10*floor(sep_100);  
octe = 10*floor(oct_100);  
nove = 10*floor(nov_100);  
dece = 10*floor(dec_100);  
  
jane1 = floor(jan_100);  
feve1 = floor(feb_100);  
mare1 = floor(mar_100);  
apre1 = floor(apr_100);  
maye1 = floor(may_100);  
june1 = floor(jun_100);  
jule1 = floor(jul_100);  
auge1 = floor(aug_100);  
sepe1 = floor(sep_100);  
octe1 = floor(oct_100);  
nove1 = floor(nov_100);  
dece1 = floor(dec_100);  
printf("Content-type: text/html\n\n");
```

jane1,feve1 etc sont des variables créées pour la fonction web afin de permettre d'afficher les % sur le graphique

```
443 printf("Content-type: text/html\n\n");
444
445 printf("<!doctype html>\n");
446 printf("<html lang=\"fr\">\n");
447 printf("<body>\n");
448
449 lecture_style();
450
451 /*
452  * Notez que pour le web que les caractères accentués ont
453  * été transformés en effet les chaines de caractères C sont nativement
454  * codées en ASCII et non en UTF-8.
455  * L'étudiant désireux de bien faire devrait normalement prétraiter
456  * tous les accents avant de les afficher en HTML.
457  */
458 printf("Depuis le 21/Fev/2018 on a enregistré %d connexions.<br>\n", nbline);
459
460 printf("<div id = \"vertgraph\">\n");
461 printf("    <ul>\n");
462 /*
463  * remarquez que chaque barre de l'histogramme est placée tous les 31 pixels,
464  * et que le label correspond ici à la hauteur/10, on n'affiche pas le label pour des
465  * hauteurs < 25 sinon ça fausse l'affichage.
466  */
467 printf("        <li style=\"left: 10px; height: %dpx;\">%d</li>\n",jane,jane1);
468 printf("        <li style=\"left: 41px; height: %dpx;\">%d</li>\n",feve,feve1);
469 printf("        <li style=\"left: 72px; height: %dpx;\">%d</li>\n",mare,mare1);
470 printf("        <li style=\"left: 103px; height: %dpx;\">%d</li>\n",apre,apre1);
471 printf("        <li style=\"left: 134px; height: %dpx;\">%d</li>\n",maye,maye1);
472 printf("        <li style=\"left: 165px; height: %dpx;\">%d</li>\n",june,june1);
473 printf("        <li style=\"left: 196px; height: %dpx;\">%d</li>\n",jule,jule1);
474 printf("        <li style=\"left: 227px; height: %dpx;\">%d</li> \n",auge,auge1);
475 printf("        <li style=\"left: 258px; height: %dpx;\">%d</li> \n",sepe,sepe1);
476 printf("        <li style=\"left: 289px; height: %dpx;\">%d</li> \n",note,note1);
477 printf("        <li style=\"left: 320px; height: %dpx;\">%d</li> \n",dece,dece1);
478 printf("    </ul>\n");
479 printf("</div>\n");
480
481 printf("</html>\n");
482 printf("</body>\n\n");
483 }
```

Nous pouvons apercevoir que nous avons modifié le heigt de chaque ligne et ce qu'il y'a entre les > < et nous y avons ajouté nos variables.

Voici la partie de l'exécution :



Désormais nous avons notre version web.

Plus qu'à finaliser le Makefile

5) Le Makefile

Nous nous sommes servis de nombreuses commandes bash pour que l'installation se fasse au mieux avec le make et le make install :

```

1 CC = gcc
2 EXEC = versiondynamique
3 LIB = -lm -lSDL -lSDL_ttf
4 LIBS= ../lib/libgraphique.c -lm -lSDL -lSDL_ttf -L/usr/lib -lm -lglib-2.0
5 EXEC= versiondynamique
6 CGIDIR = /usr/lib/cgi-bin/
7 LOGDIR = /var/log/apache2/
8 WWWDIR = /var/www/
9 HTMLDIR= $(WWWDIR)/html/
10
11 %.o: %.c
12     $(CC) $(FLAGS)-c $<
13
14 all: $(EXEC)
15
16
17 versiondynamique: versiondynamique.o
18     $(CC) $(LIBS) $(FLAGS) -o $@ $^
19
20 install: install-html install-cgi
21     @sudo chmod a+rx $(LOGDIR)
22     @sudo chmod a+r $(LOGDIR)/access.log
23
24 install-cgi:
25     @cd ../www/codescompiles/ ; \
26     sudo cp aecrire.html versiondynamique-txt \
27     $(CGIDIR) ; \
28     cd ../..
29     @echo "Installation du programme :"
30     sudo cp ../www/codescompiles/versiondynamique $(CGIDIR)
31
32 install-html:
33     @echo "Copie des fichiers aux bons emplacements"
34     @sudo tar xf ../www/www.tar -C /

```

```

36 clean:
37     @sudo rm -rf /var/www/*
38     @sudo mkdir $(HTMLDIR)
39     @sudo cp ../www/index.html $(HTMLDIR)
40     @sudo rm -f $(CGIDIR)/*
41

```

Executions :

```

user@debian: ~/SAE/SAE15/src$ make
gcc -c versiondynamique.c
gcc ../lib/libgraphique.c -lm -lSDL -lSDL_ttf -L/usr/lib -lm -lglib-2.0
-o versiondynamique versiondynamique.o
user@debian:~/SAE/SAE15/src$ make install
Copie des fichiers aux bons emplacements
Installation du programme :
sudo cp ../www/codescompiles/versiondynamique /usr/lib/cgi-bin/
user@debian:~/SAE/SAE15/src$

```

Désormais tout fonctionne, dans les dossiers tout est copié, compilé etc.

Merci d'avoir lu notre Compte Rendu,
ARVIN-BÉROD Maxence GIRAULT Adrien