Algorithmique 3 : structures de données arborescentes

Projet

Comptage exclusif de mots

Généralités

Le projet consiste à écrire un programme en C dont le but est d'afficher le nombre d'occurrences de chaque mot qui apparait dans un et un seul (de manière exclusive donc) des fichiers dont les noms figurent sur la ligne de commande.

Un mot est, par défaut, une séquence de longueur maximale de caractères n'appartenant pas à la classe `isspace`.

Les résultats sont affichés en colonnes sur la sortie standard. Les colonnes sont (uniquement) séparées par le caractère de tabulation horizontale `'\t'`. Les lignes sont (uniquement) terminées par le caractère de fin de ligne `'\n'`. Une ligne d'en-tête indique les noms des fichiers : le nom du premier fichier est affiché dans la deuxième colonne, celui du deuxième dans la troisième, etc. Pour les lignes suivantes, un mot est affiché dans la première colonne, son nombre d'occurrences dans le fichier dans lequel il apparait à l'exclusion de tous les autres dans la colonne associée au fichier. Aucun caractère de tabulation n'est affiché sur une ligne après le nombre d'occurrences.

MÀJ 12-03 Précision pour les tabulations.

L'exécutable doit être nommé `xwc` (pour *exclusive word counting*).

Exemples basiques

Trois fichiers textes sont tout d'abord créés, à savoir `x0.txt`, `x1.txt` et `x2.txt`. Trois exécutions sont ensuite lancées, sur un, deux, puis trois de ces fichiers :

```
$ cat > x0.txt
a ee iii oooo uuuuu yyyyyy
$ cat > x1.txt
a zz eee rrrr ttttt yyyyyy
$ cat > x2.txt
a bb ccc dddd eeeee ffffff ggggg hhhh iii jj k
a bb ccc dddd eeeee hhhh iii jj k
a bb ccc hhhh iii jj k
a bb ccc jj k
a jj k
a
$ ./xwc x2.txt
          ⟩x2.txt
a         ⟩6
bb        ⟩4
ccc       ⟩4
dddd      ⟩2
eeeee     ⟩2
ffffff    ⟩1
ggggg     ⟩1
hhhh      ⟩3
iii       ⟩3
jj        ⟩5
```

```
k──────⟩5
$ ./xwc x0.txt x1.txt
──────⟩x0.txt─⟩x1.txt
ee─────⟩1
iii────⟩1
oooo───⟩1
uuuuu──⟩1
zz─────⟩──────⟩1
eee────⟩──────⟩1
rrrr───⟩──────⟩1
ttttt──⟩──────⟩1
$ ./xwc x0.txt x1.txt x2.txt
──────⟩x0.txt─⟩x1.txt─⟩x2.txt
ee─────⟩1
oooo───⟩1
uuuuu──⟩1
zz─────⟩──────⟩1
eee────⟩──────⟩1
rrrr───⟩──────⟩1
ttttt──⟩──────⟩1
bb─────⟩──────⟩──────⟩4
ccc────⟩──────⟩──────⟩4
dddd───⟩──────⟩──────⟩2
eeeee──⟩──────⟩──────⟩2
ffffff─⟩──────⟩──────⟩1
ggggg──⟩──────⟩──────⟩1
hhhh───⟩──────⟩──────⟩3
jj─────⟩──────⟩──────⟩5
k──────⟩──────⟩──────⟩5
```

La première exécution ne permet de connaitre que le nombre d'occurrences de chacun des mots du fichier `x2.txt`. La deuxième affiche les nombres d'occurrences de chacun des mots en propre des fichiers `x0.txt` et `x1.txt`, ne faisant pas apparaitre les mots communs, `a` et `yyyyyy`. Même chose pour la troisième, sur les trois fichiers cette fois, avec la disparition supplémentaire du mot `iii`, commun à `x0.txt` et `x2.txt`.

Exemples avec options

Des options peuvent être ajoutées au programme, fournies sur la ligne de commande selon les canons usuels, qui rendent les productions des exécutions plus intéressantes.

La longueur maximale prise en compte pour les mots peut être fixée. Au delà de cette longueur, tout mot est coupé. La longueur est fixée à 2 dans l'exemple ci-dessous, les avertissements signifiant la coupure de mots étant redirigés vers le trou noir :

```
$ ./xwc -i2 x0.txt x1.txt x2.txt 2>/dev/null
──────⟩x0.txt─⟩x1.txt─⟩x2.txt
oo─────⟩1
uu─────⟩1
zz─────⟩──────⟩1
rr─────⟩──────⟩1
tt─────⟩──────⟩1
```

```
bb⟩⟩⟩4
cc⟩⟩⟩4
dd⟩⟩⟩2
ff⟩⟩⟩1
gg⟩⟩⟩1
hh⟩⟩⟩3
jj⟩⟩⟩5
k⟩⟩⟩5
```

Les trois fichiers ayant en partage des mots préfixés par `ee`, ces derniers disparaissent de la production.

Les résultats peuvent être ordonnés. Triés dans l'ordre lexicographique croissant des mots :

```
$ ./xwc -l x0.txt x1.txt x2.txt
⟩x0.txt⟩x1.txt⟩x2.txt
bb⟩⟩⟩4
ccc⟩⟩⟩4
dddd⟩⟩⟩2
ee⟩1
eee⟩⟩1
eeeee⟩⟩⟩2
ffffff⟩⟩⟩1
ggggg⟩⟩⟩1
hhhh⟩⟩⟩3
jj⟩⟩⟩5
k⟩⟩⟩5
oooo⟩1
rrrr⟩⟩1
ttttt⟩⟩1
uuuuu⟩1
zz⟩⟩1
```

ou dans l'ordre croissant des nombres d'occurrences, première clé, et dans l'ordre lexicographique croissant des mots, deuxième clé :

```
$ ./xwc -n x0.txt x1.txt x2.txt
⟩x0.txt⟩x1.txt⟩x2.txt
ee⟩1
eee⟩⟩1
ffffff⟩⟩⟩1
ggggg⟩⟩⟩1
oooo⟩1
rrrr⟩⟩1
ttttt⟩⟩1
uuuuu⟩1
zz⟩⟩1
dddd⟩⟩⟩2
eeeee⟩⟩⟩2
hhhh⟩⟩⟩3
bb⟩⟩⟩4
ccc⟩⟩⟩4
jj⟩⟩⟩5
k⟩⟩⟩5
```

Ou, inversement, dans l'ordre lexicographique décroissant des mots :

```
$ ./xwc -l -R x0.txt x1.txt x2.txt
──────⟩x0.txt─⟩x1.txt─⟩x2.txt
zz────⟩──────⟩1
uuuuu─⟩1
ttttt─⟩──────⟩1
rrrr──⟩──────⟩1
oooo──⟩1
k─────⟩──────⟩──────⟩5
jj────⟩──────⟩──────⟩5
hhhh──⟩──────⟩──────⟩3
ggggg─⟩──────⟩──────⟩1
ffffff⟩──────⟩──────⟩1
eeeee─⟩──────⟩──────⟩2
eee───⟩──────⟩1
ee────⟩1
dddd──⟩──────⟩──────⟩2
ccc───⟩──────⟩──────⟩4
bb────⟩──────⟩──────⟩4
```

ou dans l'ordre décroissant des nombres d'occurrences, première clé, et dans l'ordre lexicographique croissant des mots, deuxième clé :

```
$ ./xwc -n -R x0.txt x1.txt x2.txt
──────⟩x0.txt─⟩x1.txt─⟩x2.txt
jj────⟩──────⟩──────⟩5
k─────⟩──────⟩──────⟩5
bb────⟩──────⟩──────⟩4
ccc───⟩──────⟩──────⟩4
hhhh──⟩──────⟩──────⟩3
dddd──⟩──────⟩──────⟩2
eeeee─⟩──────⟩──────⟩2
ee────⟩1
eee───⟩──────⟩1
ffffff⟩──────⟩──────⟩1
ggggg─⟩──────⟩──────⟩1
oooo──⟩1
rrrr──⟩──────⟩1
ttttt─⟩──────⟩1
uuuuu─⟩1
zz────⟩──────⟩1
```

Une restriction peut être imposée aux mots à traiter : appartenir en plus à l'ensemble des mots qui figurent dans un fichier texte donné pour l'occasion.

```
$ ./xwc -r x0.txt x1.txt x2.txt
x0.txt─⟩x1.txt─⟩x2.txt
iii───⟩──────⟩3
yyyyyy─⟩1
```

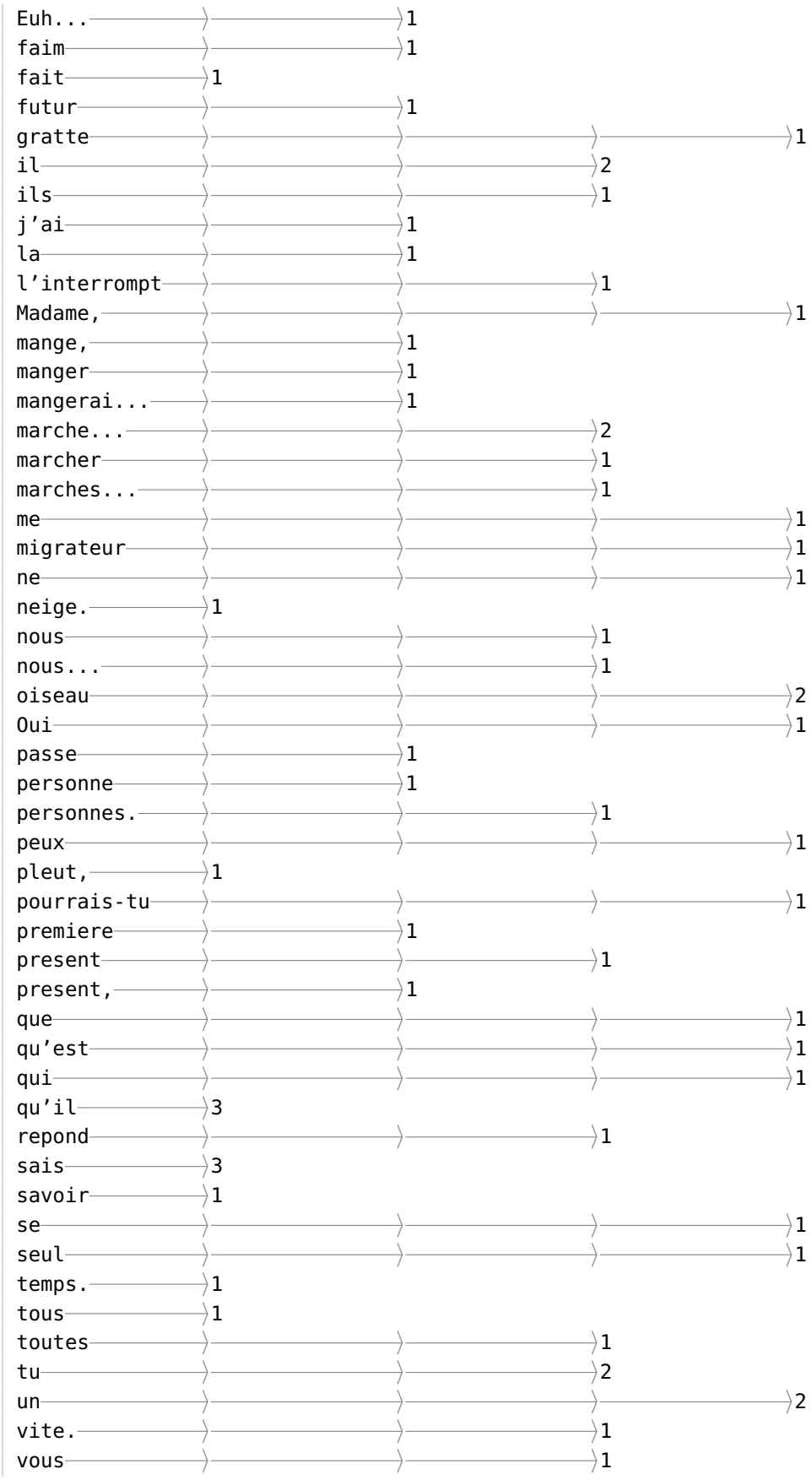
Le nom du « fichier restricteur » est affiché dans la première colonne de la ligne d'en-tête.

Sur des textes plus « réels » maintenant, en ce sens où y figurent des mots de langue française et des symboles de ponctuation :

```
$ cat > toto0.txt
La maitresse de Toto lui demande :
- Toto, conjugue-moi le verbe savoir a tous les temps.
- Je sais qu'il pleut, je sais qu'il fait beau, je sais qu'il neige.
$ cat > toto1.txt
La maitresse demande a Toto de conjuguer le verbe manger a la premiere personne
 du present, du futur et du passe compose.
Toto dit :
- Euh... je mange, je mangerai... euh... j'ai plus faim !
$ cat > toto2.txt
La maitresse demande a Toto de conjuguer le verbe marcher au present et a toutes
 les personnes.
Toto lui repond :
- Je marche... tu marches... il marche... nous...
La maitresse l'interrompt et lui demande d'aller plus vite.
Alors Toto accelere :
- Je cours, tu cours, il court, nous courons, vous courez, ils courent !
$ cat > toto3.txt
La maitresse demande a Toto :
- Toto, pourrais-tu me dire ce qu'est un oiseau migrateur ?
- Oui Madame, je peux : c'est un oiseau qui ne se gratte que d'un seul cote.
```

L'exécution sur ces quatre fichiers dans l'ordre lexicographique croissant des mots donne :

```
$ ./xwc -l toto0.txt toto1.txt toto2.txt toto3.txt
                ⟩toto0.txt⟩toto1.txt⟩toto2.txt⟩toto3.txt
?               ⟩         ⟩         ⟩         ⟩1
accelere        ⟩         ⟩         ⟩1
Alors           ⟩         ⟩         ⟩1
au              ⟩         ⟩         ⟩1
beau,           ⟩1
ce              ⟩         ⟩         ⟩         ⟩1
c'est           ⟩         ⟩         ⟩         ⟩1
compose.        ⟩         ⟩1
conjugue-moi    ⟩1
cote.           ⟩         ⟩         ⟩         ⟩1
courent         ⟩         ⟩         ⟩1
courez,         ⟩         ⟩         ⟩1
courons,        ⟩         ⟩         ⟩1
cours,          ⟩         ⟩         ⟩2
court,          ⟩         ⟩         ⟩1
d'aller         ⟩         ⟩         ⟩1
dire            ⟩         ⟩         ⟩         ⟩1
dit             ⟩         ⟩1
du              ⟩         ⟩3
d'un            ⟩         ⟩         ⟩         ⟩1
euh...          ⟩         ⟩1
```


Euh...1
faim1
fait1
futur1
gratte1
il2
ils1
j’ai1
la1
l’interrompt1
Madame,1
mange,1
manger1
mangerai...1
marche...2
marcher1
marches...1
me1
migrateur1
ne1
neige.1
nous1
nous...1
oiseau2
Oui1
passe1
personne1
personnes.1
peux1
pleut,1
pourrais-tu1
premiere1
present1
present,1
que1
qu’est1
qui1
qu’il3
repond1
sais3
savoir1
se1
seul1
temps.1
tous1
toutes1
tu2
un2
vite.1
vous1

Les mots sont bien ordonnés dans l'ordre lexicographique de la langue, avec `Alors` entre `accelere` et `au`, `ce` avant `c'est`, successivement `que`, `qu'est`, `qui` et `qu'il`. Aucun mystère à cela au niveau du codage : il suffit, après avoir fait appel en début d'exécution à la fonction `setlocale` de l'en-tête standard `<locale.h>` avec comme premier paramètre la macroconstante `LC_COLLATE` définie dans le même en-tête et comme second paramètre la chaine vide, d'utiliser la fonction de comparaison `strcoll` de l'en-tête standard `<string.h>` en lieu et place de la traditionnelle fonction `strcmp`.

Des résultats sensiblement plus cohérents peuvent toutefois être obtenus en tentant de se rapprocher de la notion de « mot » du langage en supprimant les symboles de ponctuation de la définition des mots :

```
$ ./xwc -p -l toto0.txt toto1.txt toto2.txt toto3.txt
                ⟩toto0.txt⟩toto1.txt⟩toto2.txt⟩toto3.txt
accelere        ⟩         ⟩         ⟩1
ai              ⟩         ⟩1
aller           ⟩         ⟩         ⟩1
Alors           ⟩         ⟩         ⟩1
au              ⟩         ⟩         ⟩1
beau            ⟩1
c               ⟩         ⟩         ⟩         ⟩1
ce              ⟩         ⟩         ⟩         ⟩1
compose         ⟩         ⟩1
conjugue        ⟩1
cote            ⟩         ⟩         ⟩         ⟩1
courent         ⟩         ⟩         ⟩1
courez          ⟩         ⟩         ⟩1
courons         ⟩         ⟩         ⟩1
cours           ⟩         ⟩         ⟩2
court           ⟩         ⟩         ⟩1
dire            ⟩         ⟩         ⟩         ⟩1
dit             ⟩         ⟩1
du              ⟩         ⟩3
est             ⟩         ⟩         ⟩         ⟩2
euh             ⟩         ⟩1
Euh             ⟩         ⟩1
faim            ⟩         ⟩1
fait            ⟩1
futur           ⟩         ⟩1
gratte          ⟩         ⟩         ⟩         ⟩1
ils             ⟩         ⟩         ⟩1
interrompt      ⟩         ⟩         ⟩1
j               ⟩         ⟩1
l               ⟩         ⟩         ⟩1
la              ⟩         ⟩1
Madame          ⟩         ⟩         ⟩         ⟩1
mange           ⟩         ⟩1
manger          ⟩         ⟩1
mangerai        ⟩         ⟩1
marche          ⟩         ⟩         ⟩2
marcher         ⟩         ⟩         ⟩1
```

```
marches──────⟩──────⟩──────⟩1
me───────────⟩──────⟩──────⟩──────⟩1
migrateur────⟩──────⟩──────⟩──────⟩1
moi──────────⟩1
ne───────────⟩──────⟩──────⟩──────⟩1
neige────────⟩1
nous─────────⟩──────⟩──────⟩2
oiseau───────⟩──────⟩──────⟩──────⟩2
Oui──────────⟩──────⟩──────⟩──────⟩1
passe────────⟩──────⟩1
personne─────⟩──────⟩1
personnes────⟩──────⟩──────⟩1
peux─────────⟩──────⟩──────⟩──────⟩1
pleut────────⟩1
pourrais─────⟩──────⟩──────⟩──────⟩1
premiere─────⟩──────⟩1
que─────────⟩──────⟩──────⟩──────⟩1
qui──────────⟩──────⟩──────⟩──────⟩1
repond───────⟩──────⟩──────⟩1
sais─────────⟩3
savoir───────⟩1
se───────────⟩──────⟩──────⟩──────⟩1
seul─────────⟩──────⟩──────⟩──────⟩1
temps────────⟩1
tous─────────⟩1
toutes───────⟩──────⟩──────⟩1
un───────────⟩──────⟩──────⟩──────⟩3
vite─────────⟩──────⟩──────⟩1
vous─────────⟩──────⟩──────⟩1
```

Les « ? », « ' », « - », « . », « , » et donc « ... » disparaissent, révélant les `ai`, `aller` et `c`, confondant les `nous` (`nous` et `nous...` dans `toto2.txt`), mais faisant également disparaitre le `tu` (`tu` dans `toto2.txt` et `pourrais-tu` dans `toto3.txt`).

Enfin, il peut être intéressant de traiter de petits exemples « en direct », en permettant que la lecture ait lieu directement depuis l'entrée standard. Comme dans la commande `cat` : *With no FILE, or when FILE is -, read standard input.* Dans de tels cas, « "" » est affiché en lieu et place du nom de fichier manquant.

MÀJ 12-03 Ajout de « "" ».

```
$ ./xwc -l - -
a ee iii oooo uuuuu yyyyyy[entrée] [ctrl]+[d]
a zz eee rrrr ttttt yyyyyy[entrée] [ctrl]+[d]
──────────⟩""──────⟩""
ee────────⟩1
eee───────⟩──────⟩1
iii───────⟩1
oooo──────⟩1
rrrr──────⟩──────⟩1
ttttt─────⟩──────⟩1
uuuuu─────⟩1
zz────────⟩──────⟩1
```

Mise en œuvre

Votre programme doit être écrit en C, doit pouvoir être compilé avec `gcc` et doit supporter les options de compilation usuelles : `-std=c2x`, `-Wall`, `-Wconversion`, `-Werror`, `-Wextra`, `-Wpedantic`, `-Wwrite-strings` et `-O2`.

Vos sources C doivent être mis en forme par `uncrustify` selon le fichier de configuration déjà fourni. Aucune ligne ne doit avoir une longueur strictement supérieure à 80.

Les types, sous-programmes et paramètres doivent être correctement nommés. Toute duplication de code est à éviter. En particulier, les éventuels caractères, chaines et nombres magiques seront nommés.

Vous éviterez l'utilisation de variables globales.

L'exécutable doit être nommé `xwc`.

À la base de votre travail, vous utiliserez nécessairement soit l'en-tête `"hashtable.h"`, soit l'en-tête `"bst.h"`, sans les modifier. La partie implantation associée à l'en-tête `"bst.h"` se doit de traiter les arbres binaires de recherche comme des AVL.

Vous pouvez développer et utiliser des modules additionnels, par exemple pour gérer les options ou la partie récupération de mots dans un flot texte.

Toutes les zones mémoires explicitement allouées doivent être désallouées avant la fin de l'exécution, même en cas d'erreur. Jamais deux chaines de caractères de même valeur ne doivent être allouées.

Tout message d'avertissement ou d'erreur doit être envoyé vers la sortie erreur.

Une option courte ou une option longue doit obligatoirement être supportée : « `-?` » ou « `--help` », qui affiche l'aide selon le format usuel ainsi que les éventuelles limitations de l'exécutable.

Il est possible de ne développer que des options courtes ou que des options longues.

Les options suivantes, courtes ou longues, peuvent être développées :

`-a` : même chose que `--words-processing=avl-binary-tree` ;

`-h` : même chose que `--words-processing=hash-table` ;

`-w TYPE` ou `--words-processing=TYPE` : pour signifier le traitement des mots selon la structure de données spécifiée par `TYPE`. Les valeurs possibles pour `TYPE` sont explicites : `avl-binary-tree` et `hash-table`. La valeur par défaut de `TYPE` est `hash-table` ;

`-i VALUE` ou `--initial=VALUE` : pour fixer la longueur maximale des mots à `VALUE`. Lorsque `VALUE` vaut 0, la longueur n'a aucune limite. La valeur par défaut de `VALUE` est 0 ;

`-p` ou `--punctuation-like-space` : pour donner aux symboles de ponctuation le même rôle que les caractères de la classe `isspace` ;

`-r FILE` ou `--restrict=FILE` : pour limiter le comptage à l'ensemble des mots qui apparait dans le fichier signifié par `FILE`. `FILE` apparait à la première colonne de la ligne en-tête.

`-l` : même chose que `--sort=lexicographical` ;

`-n` : même chose que `--sort=numeric` ;

`-S` : même chose que `--sort=none` ;

MÀJ 07-04 `-S` auparavant oubliée.

`-s TYPE` ou `--sort=TYPE` : pour trier le résultat dans l'ordre croissant, par défaut, selon `TYPE`. Les valeurs possibles pour `TYPE` sont : `lexicographical`, tri sur les mots, `numeric`, tri sur le nombre d'occurrences, clé primaire, et sur les mots, clé secondaire, et `none`, sans essayer de trier, en prenant les éléments comme ils viennent. La valeur par défaut de `TYPE` est `none` ;

`-R` ou `--reverse` : pour trier dans l'ordre décroissant sur la seule ou la première des clés au lieu du tri dans l'ordre croissant. Cette option est sans effet lorsque l'option `-S` est active.

À l'instar de certaines commandes Linux, la possibilité de lire les mots sur l'entrée standard au lieu de les lire dans un fichier de nom donné sur la ligne de commande peut être développée. Pour le signifier à la commande, il suffit d'utiliser « `-` » en lieu et place d'un nom de fichier. Dans l'affichage de la ligne en-tête, « `""` » doit figurer à l'emplacement du nom de fichier manquant.

MÀJ 12-03 Ajout de « `""` ».

Le fait qu'aucun nom de fichier ne figure sur la ligne de commande peut aussi ne pas donner lieu à un message d'erreur. Il convient lorsque cette possibilité est offerte de se rabattre sur l'entrée standard, exactement comme si « `-` » figurait sur la ligne de commande.

Modalités

Vous pouvez réaliser ce projet à deux, pas plus.
Vous soutiendrez le projet individuellement.

Votre projet sera compilable et exécutable sur les machines des salles de travaux pratiques du département d'informatique et sous Xubuntu.

Votre programme devra répondre aux diverses exigences signifiées dans la section « Mise en œuvre ». Il sera en partie testé avec des outils automatiques.

Votre projet devra être rendu avec tous les fichiers sources nécessaires, un fichier `makefile`, un rapport de développement. Le rapport sera fourni sous forme électronique (fichier au format `pdf`) qui précisera l'objet de chacun des modules, décrira l'implantation (dessins souhaités), commentera des exemples, explicitera les éventuelles limitations. Aucun autre fichier que ceux-là ne doit être joint ; en particulier, aucun des textes ayant servi aux exemples.

Vous remettrez votre projet au plus tard le **lundi 13 mai 2024 à 18 h 47** dans une archive au format `tar.gz` de nom *identifiant*`_projet.tar.gz`, où *identifiant* est l'identifiant de 8 caractères de l'un des membres du groupe. Cette archive sera envoyée en pièce jointe à un courriel adressé à l'ensemble des chargé·es de TP et de sujet « Algo3 projet ». Un seul envoi sera effectué par groupe : l'éventuel autre membre du groupe sera mis en copie du message.

MÀJ 17-04 Fixation de la date de remise.

La notation sera sensible à la propreté du code, à son homogénéité, aux performances de l'exécutable, à la clarté des explications fournies dans le rapport et lors de la soutenance, ainsi qu'à un passage sans encombre au révélateur `valgrind`.

Hors jeux

La note attribuée au projet est 0/20 en cas :

— de demande expresse de la part de l'étudiant·e,

MÀJ 12-03 Ajout.

— d'erreur d'envoi : destinataires, sujet,
— d'erreur d'identifiant d'archive,
— d'archive non conforme : format, contenu (fichier manquant, tel un source, le fichier `makefile` ou le rapport ; fichier en trop, tel un exécutable, un fichier objet, un fichier temporaire),

MÀJ 07-04 Précision.

— d'erreur apparaissant à compilation,
— de non respect de la mise en forme standard,
— pour toute recopie dans un source quelconque d'une structure normalement cachée dans la partie implantation d'un module,
— pour tout plagiat (recopie totale ou partielle d'un source tiers),

MÀJ 07-04 Précision.

— pour méconnaissance du projet présenté lors de la soutenance.

Grille d'évaluation

MÀJ 07-04 Ajout de la section.

En dehors des cas rédhibitoires précédemment cités, la réalisation du développement et de sa présentation est notée sur un maximum de 25 points (ramenés à 20 en cas d'excès) répartis comme suit :

MÀJ 10-04 Mise à jour de la distribution des points.

critère	*points*
pour 2 fichiers uniquement	7
pour un nombre de fichiers supérieur ou égal à 1 uniquement	11
pour un nombre quelconque de fichiers, dont aucun fichier (entrée standard donc) et un seul fichier	12
options correspondant à `-a` et `-h`	2
option `-i`	1
option `-p`	0,5
option `-r`	1
option correspondant à `-l`	1
option correspondant à `-n`	1
option `-R`	0,5
utilisation de « `-` »	1
rapport et soutenance	5

Des pourcentages de points peuvent être retirés en fonction des critères suivants :

critère	*pénalités*
absence d'aide ou aide en inadéquation avec ce qui est développé	−20 %
plantage dans la phase de reconnaissance des options (une option non supportée doit donner lieu à un message d'erreur et à l'arrêt de l'exécutable)	−20 %
gestion non correcte d'opérations sur les fichiers ou plantage consécutif à cette mauvaise gestion (toutes les opérations sur le fichiers doivent être testées et correctement traitées)	−30 %
erreurs signalées par `valgrind`	−50 %
désallocations non faites signalées par `valgrind`	−25 %
messages d'erreur inadaptés (les textes des messages d'erreur doivent décrire les erreurs de la manière la plus appropriée possible et doivent être envoyés vers la sortie erreur)	−10 %
complexités non satisfaisantes à l'exécution	−25 %
utilisation de variables globales (hors `errno`)	−10 %
présence de nombres ou chaines magiques	−10 %
répétitions de parties de code (le code doit être factorisé, les fonctions auxiliaires doivent être spécifiées)	−20 %
code mal présenté, malgré l'utilisation de `uncrustify`	−25 %

MÀJ 21-04 or → hors (CG)

Foire aux questions

► 1 Pour les options. Elles sont avant les fichiers ? ou alors elles peuvent être n'importe où sur la ligne de commande ?

Rien ne dit dans la syntaxe « `command [OPTION]... [FILE]...` » qu'options et (noms de) fichiers (ou assimilés) peuvent être entremêlés. Cependant, lorsque la syntaxe de la commande le permet, il est très agréable du point de vue utilisateurice de faire n'importe quoi… Comme par exemple de taper « `-?` » ou « `--help` » en dernier sur la ligne de commandes. Cela dit, ça peut ne pas fonctionner à chaque fois :

```
$ diff --nimportequoi --help
diff : l'option « --nimportequoi » n'a pas été reconnue
diff: Utilisez « diff --help » pour en savoir d'avantage.
```

Pour que ça fonctionne, il aurait fallut que ce qui précède sur la ligne de commande soit syntaxiquement correct

Dans la version de `xwc` que j'ai déposée sur UniversiTICE, options et fichiers peuvent être entremêlés. Des formes réduites des options courtes peuvent également être employées. Par exemple « `./xwc -p -l ...` » peut tout aussi bien être formulé « `./xwc -pl ...` » Vous n'êtes en aucun cas tenu de faire de même.

Pour vous rassurez définitivement : le test sur l'aide sera fait « à vide », soit « `./xwc -?` » ou « `./xwc --help` » ; les autres tests ne seront faits qu'avec les éventuelles options en premier.

► 2 L'information est peut être dans le sujet mais si c'est le cas je n'ai pas réussi à la trouver : est-il nécessaire de conserver l'ordre de lecture des mots lors de l'affichage des compteurs si aucune option de tri n'est donnée en entrée ? Dans les différents exemples et les tests de xwc que j'ai réalisé, l'ordre des mots est le même que celui du texte ; quand il y a plusieurs textes, les mots sont d'abord ceux du premier fichier, puis du second…

Non, il n'est pas nécessaire de conserver l'ordre de lecture si aucun tri n'est demandé par l'utilisateurice. Les « éléments » (les mots et leurs nombres d'occurrences) sont pris « comme ils viennent », c'est-à-dire tels qu'ils sont mis à disposition par la structure de données. Clairement, l'utilisation d'un fourretout avec insertion en queue amène au résultat affiché par l'exécutable fourni. À moins que ce ne soit celle de deux fourretouts avec insertion en tête, l'un des deux étant renversé dans second…

► 3 Est ce que la note de projet remplace celle de [l'examen de] TP si elle lui est supérieur comme pour les CC ?

C'est le contraire. Voir « Annonces / SEA 19… » sur UniversiTICE.

► 4 Est ce qu'on a le droit de réutiliser exactement le même `holdall.c/.h` et tout les autres fichier qu'on a vu lors des TP dans nos projets ?

Oui. En détail : si vous utilisez le module `holdall`, vous ne pouvez modifier de `"holdall.h"` que la ligne mentionnée dans sa spécification ; si vous utilisez le module `hashtable`, vous ne pouvez pas modifier `"hashtable.h"` ; si vous utilisez le module `bst`, vous ne pouvez pas modifier `"bst.h"`. À titre indicatif, dans la version de `xwc` que j'ai déposée sur UniversiTICE : je n'ai pas modifié `holdall.c` que pour ajouter le tri (voir fiche de TP n° 1) ; j'ai donc modifié `"holdall.h"` en conséquence ; je n'ai pas modifié `hashtable.c` ; j'ai utilisé `bst.c` comme j'ai pu le confectionner à l'issue du travail sur la fiche de TP n° 5. Si cela vous chante et si vous en avez le temps, vous pouvez revoir les parties implantations de ces modules.

► 5 Par rapport a l'implentation des AVL et table de hachage est ce que le fait de faire les deux rentre dans le barème, ou est ce que ce sont des points "bonus" ?

Points bonus. Voir grille d'évaluation.

▶ 6 Vous nous avez donner des contraintes sur le temps d'éxecution explicitement en CM, mais en terme d'espace qu'on utilise on fait ce qu'on veut ou ? Comme déclarer 5 tables de hachages par exemple toutes de la taille du texte le plus long (je sais il y a aucune raison de le faire mais par exemple)

Faites comme vous pouvez. Il vous faudra toutefois être capable de justifier votre choix, dans votre rapport comme lors de la soutenance.

▶ 7 Si jamais pour une quelquonque raison sur les pc de la fac notre projet ne fonctionne pas, est ce qu'on est autorisé à soutenir avec nos machines personnels ?

Non. Le juge arbitre sont les machines des salles de travaux pratiques du département d'informatique. Avec les options de compilation usuelles, et uniquement celles-là. Voir « Mise en œuvre » et « Modalités ».

▶ 8 Si le programme fonctionne mais que le temps d'éxécution est très élevé, on se confronterait à un malus de combien de points environ ?

Voir grille d'évaluation.

▶ 9 Même question pour`valgrind`, si on a 1000000000 allocation pour 10 free par exemple.

Voir grille d'évaluation.

▶ 10 Est ce que les points seront partagé entre le code qu'on fournie et notre prestation lors de la soutenance ?

Voir grille d'évaluation.

▶ 11 Si l'on a fais uniquement le travail minimum, c'est à dire le compteur de mot exclusif sans aucune option sur combien de point on est noté ?

Voir grille d'évaluation.

▶ 12 Devrons nous débattre sur les choix d'implentation qu'on a fait ou seulement répondre à des questions sur notre code ?

Pas « débattre » mais « justifier ». En plus d'une « animation » au tableau sur un exemple et de questions sur le code.

▶ 13 Rassurez-moi : ce n'est pas bon si j'utilise un tableau pour les nombres d'occurrences comme dans la fiche de TP numéro 1 ?

Le problème, c'est la limitation des compteurs des nombres d'occurrences. Donc, non, ce n'est pas bon. Cela dit, si certain·es étaient intéressé·es, une possibilité, puis une option par la suite, pourrait être ajoutée pour envisager une telle limitation.

▶ 14 J'ai deux fois plus d'allocations que vous. Est-ce que je serai sanctionnée ?

Non, du moment que l'exécutable est performant. Cela dit : l'important n'est pas le nombre d'allocations mais l'espace total alloué ; il reste du temps — aujourd'hui : 26 mars — pour éventuellement pour réduire ce nombre d'allocations.

▶ 15 Par rapport à l'option [courte `-R` et longue] `--reverse`, je voulais demander plus d'explications sur son fonctionnement. En essayant de l'éxecuter, je ne remarque aucune difference à l'affichage.

```
$ cat > x3.txt
a a a a ee iii oooo uuuuu yyyyyy e a a a a ee e a a a a a ee e e a a a a a ee e
e a a a a a ee e e a a a a a ee e e
$ ./xwc -R x3.txt
————————⟩x3.txt
a————————⟩28
```

```
ee──────⟩6
iii─────⟩1
oooo────⟩1
uuuuu───⟩1
yyyyyy──⟩1
e───────⟩10
```

Cependant, en rajoutant `n` (`l` respectivement) je constate que l'affichage est bien trié dans l'ordre décroissant numérique (lexicographique respectivement).

```
$ ./xwc -Rl x3.txt
──────────⟩x3.txt
yyyyyy──⟩1
uuuuu───⟩1
oooo────⟩1
iii─────⟩1
ee──────⟩6
e───────⟩10
a───────⟩28
```

Mais en mettant `n` et `l` en meme temps (pour avoir `n` en clé primaire et `l` en clé secondaire par exemple), je constate que l'affichage est trié dans l'ordre décroissant de la clé secondaire contrairement à ce qui est affirmé dans le sujet.

```
$ ./xwc -Rln x3.txt
──────────⟩x3.txt
a───────⟩28
e───────⟩10
ee──────⟩6
iii─────⟩1
oooo────⟩1
uuuuu───⟩1
yyyyyy──⟩1
```

Pour la première exécution : comme spécifié, `-R` « est sans effet lorsque l'option `-S` est active » ; donc par défaut.

Pour la troisième exécution : `-Rln` ne signifie pas « `n` en clé primaire et `l` en clé secondaire » mais « 1) s'il est demandé que la production soit triée, le faire dans l'ordre décroissant pour le tri à une seule clé — lexicographique donc — et dans l'ordre décroissant sur la première clé pour le tri à deux clés — numérique donc — et dans l'ordre croissant sur la deuxième clé ; 2) effectuer un tri lexicographique ; 3) tiens, et puis non, j'ai changé d'avis : effectuer un tri numérique. » De manière générale, pour chaque option donnée, c'est le dernier argument donné qui l'emporte. Voici, en définitive, un tri lexicographique dans l'ordre décroissant :

```
$ ./xwc -Rlnlnlnlnlnl x3.txt
──────────⟩x3.txt
yyyyyy──⟩1
uuuuu───⟩1
oooo────⟩1
iii─────⟩1
ee──────⟩6
e───────⟩10
a───────⟩28
```

un tri numérique dans l'ordre décroissant :

```
$ ./xwc -Rlnlnlnlnlnln x3.txt
————————⟩x3.txt
a———————⟩28
e———————⟩10
ee——————⟩6
iii—————⟩1
oooo————⟩1
uuuuu———⟩1
yyyyyy——⟩1
```

un tri lexicographique dans l'ordre décroissant :

```
$ ./xwc -S -Rl -Rn -S -Rl -l x3.txt
————————⟩x3.txt
yyyyyy——⟩1
uuuuu———⟩1
oooo————⟩1
iii—————⟩1
ee——————⟩6
e———————⟩10
a———————⟩28
```

et un tri numérique dans l'ordre décroissant :

```
$ ./xwc -S -Rl -Rn -S -Rl -n x3.txt
————————⟩x3.txt
a———————⟩28
e———————⟩10
ee——————⟩6
iii—————⟩1
oooo————⟩1
uuuuu———⟩1
yyyyyy——⟩1
```

► 16 Mon exécutable fonctionne parfaitement bien. Mais `valgrind` m'indique plein d'erreurs, d'autant plus que la taille du fichier est importante.

```
$ valgrind ./xwc lesmiserables.txt
...
==5020==
==5020== HEAP SUMMARY:
==5020==     in use at exit: 0 bytes in 0 blocks
==5020==   total heap usage: 180,760 allocs, 180,760 frees, 5,353,771 bytes allo
cated
...
==5020== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
```

D'où ça vient ?

Donc l'exécutable ne fonctionne pas parfaitement bien... Il manque une ligne importante à la citation de la production de `valgrind`, celle qui, ici, précède la dernière ligne :

```
==5020== For lists of detected and suppressed errors, rerun with: -s
==5020== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
```

Avec `-s` donc :

```
$ valgrind ./xwc lesmiserables.txt
...
==5307==
==5307== HEAP SUMMARY:
==5307==     in use at exit: 0 bytes in 0 blocks
==5307==   total heap usage: 180,760 allocs, 180,760 frees, 5,353,771 bytes allo
cated
...
==5307==
==5307== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
==5307==
==5307== 883 errors in context 1 of 2:
==5307== Invalid read of size 1
==5307==    at 0x484D000: strcmp (in /usr/libexec/valgrind/...)
==5307==    by 0x1098E8: hashtable__search (in .../xwc)
==5307==    by 0x109C6C: hashtable_search (in .../xwc)
==5307==    by 0x109538: main (in .../xwc)
==5307==  Address 0x52bf3e1 is 1 bytes inside a block of size 11 free'd
==5307==    at 0x484810F: free (in /usr/libexec/valgrind/...)
==5307==    by 0x109567: main (in .../xwc)
==5307==  Block was alloc'd at
==5307==    at 0x4845828: malloc (in /usr/libexec/valgrind/...)
==5307==    by 0x1093FD: main (in .../xwc)
==5307==
==5307==
==5307== 13235 errors in context 2 of 2:
==5307== Invalid read of size 1
==5307==    at 0x484CFE7: strcmp (in /usr/libexec/valgrind/...)
==5307==    by 0x1098E8: hashtable__search (in .../xwc)
==5307==    by 0x109C6C: hashtable_search (in .../xwc)
==5307==    by 0x109538: main (in .../xwc)
==5307==  Address 0x4ec0170 is 0 bytes inside a block of size 10 free'd
==5307==    at 0x484810F: free (in /usr/libexec/valgrind/...)
==5307==    by 0x109567: main (in .../xwc)
==5307==  Block was alloc'd at
==5307==    at 0x4845828: malloc (in /usr/libexec/valgrind/...)
==5307==    by 0x1093FD: main (in .../xwc)
==5307==
==5307== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
```

Et si, le temps de la phase de tests, sont données à `gcc` les options « `-O0 -g` » au lieu de « `-O2` », `valgrind` fourni les numéros de lignes :

```
$ valgrind ./xwc lesmiserables.txt
...
==5525==
==5525== HEAP SUMMARY:
==5525==     in use at exit: 0 bytes in 0 blocks
```

```
==5525==   total heap usage: 180,760 allocs, 180,760 frees, 5,353,771 bytes allo
cated
...
==5525==
==5525== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
==5525==
==5525== 883 errors in context 1 of 2:
==5525== Invalid read of size 1
==5525==    at 0x484D000: strcmp (in /usr/libexec/valgrind/...)
==5525==    by 0x109A4D: hashtable__search (hashtable.c:97)
==5525==    by 0x109FD9: hashtable_search (hashtable.c:236)
==5525==    by 0x109777: main (main.c:161)
==5525==  Address 0x52bf3e1 is 1 bytes inside a block of size 11 free'd
==5525==    at 0x484810F: free (in /usr/libexec/valgrind/...)
==5525==    by 0x1097D9: main (main.c:164)
==5525==  Block was alloc'd at
==5525==    at 0x4845828: malloc (in /usr/libexec/valgrind/...)
==5525==    by 0x1095C6: main (main.c:113)
==5525==
==5525==
==5525== 13235 errors in context 2 of 2:
==5525== Invalid read of size 1
==5525==    at 0x484CFE7: strcmp (in /usr/libexec/valgrind/...)
==5525==    by 0x109A4D: hashtable__search (hashtable.c:97)
==5525==    by 0x109FD9: hashtable_search (hashtable.c:236)
==5525==    by 0x109777: main (main.c:161)
==5525==  Address 0x4ec0170 is 0 bytes inside a block of size 10 free'd
==5525==    at 0x484810F: free (in /usr/libexec/valgrind/...)
==5525==    by 0x1097D9: main (main.c:164)
==5525==  Block was alloc'd at
==5525==    at 0x4845828: malloc (in /usr/libexec/valgrind/...)
==5525==    by 0x1095C6: main (main.c:113)
==5525==
==5525== ERROR SUMMARY: 14118 errors from 2 contexts (suppressed: 0 from 0)
```

Clairement donc, des erreurs lors d'opérations d'allocation dynamique.

► 17 Si dans les fichiers il y a des mots qui se ne distinguent que par le fait que certaines de leurs lettres sont en capitales dans les uns et en minuscules dans les autres, comme `azerty`, `Azerty`, `AzErTy` ou `AZERTY`, ça doit être considéré comme des mots identiques ou des mots différents ?

Les quatre mots cités sont différents. Ils doivent donc être comptés chacun de son côté. C'est ensuite à `strcoll` de jouer si un tri est demandé. Par exemple :

```
$ ./xwc - --sort=lexicographical
--- starts reading for #1 FILE
azerty Azerty AzErTy AZERTY qsdfgh Qsdfgh qSdFgH QSDFGH xwc xWC Xwc XWC IO Io io
azerty        AzErTy               Qsdfgh        QSDFGH xwc             IO Io io
              AzErTy               Qsdfgh               xwc             IO    io
                     AZERTY                                     Xwc     IO
```

```
--- ends reading for #1 FILE
──────┤""
azerty┤2
Azerty┤1
AzErTy┤3
AZERTY┤2
io────┤3
Io────┤2
IO────┤4
qsdfgh┤1
qSdFgH┤1
Qsdfgh┤3
QSDFGH┤2
xwc───┤3
xWC───┤1
Xwc───┤2
XWC───┤1
```

► **18 Et s'il y a des lettres accentuées dans les mots ? Il faut en tenir compte ?**

Ne vous en souciez pas. Lisez avec `fgetc`, assemblez dans un tableau de **`char`**, affichez avec `fprintf`. Par exemple :

```
$ ./xwc - --punctuation-like-space --sort=lexicographical
--- starts reading for #1 FILE
Au bout de quelques secondes, la chambre et le mur d'en face furent
éclairés d'une grande réverbération rouge et tremblante. Tout brûlait.
Le bâton d'épine pétillait et jetait des étincelles jusqu'au milieu de
la chambre.
--- ends reading for #1 FILE
──────┤""
au────┤1
Au────┤1
bâton──┤1
bout──┤1
brûlait┤1
chambre┤2
d─────┤3
de────┤2
des───┤1
éclairés──────┤1
en────┤1
épine─┤1
et────┤3
étincelles────┤1
face──┤1
furent┤1
grande┤1
jetait┤1
jusqu─┤1
la────┤2
```

```
le⟶1
Le⟶1
milieu⟶1
mur⟶1
pétillait⟶1
quelques⟶1
réverbération⟶1
rouge⟶1
secondes⟶1
Tout⟶1
tremblante⟶1
une⟶1
```

C'est en conséquence plus frustrant dès lors qu'une longueur maximale est fixée pour les mots, les lettres accentuées ou cédillées étant codées sur deux *bytes*. Par exemple :

```
$ ./xwc - --punctuation-like-space --sort=lexicographical --initial=3
--- starts reading for #1 FILE
Au bout de quelques secondes, la chambre et le mur d'en face furent
xwc: Word from standard input at line 1 cut: 'bou...'.
xwc: Word from standard input at line 1 cut: 'que...'.
xwc: Word from standard input at line 1 cut: 'sec...'.
xwc: Word from standard input at line 1 cut: 'cha...'.
xwc: Word from standard input at line 1 cut: 'fac...'.
xwc: Word from standard input at line 1 cut: 'fur...'.
éclairés d'une grande réverbération rouge et tremblante. Tout brûlait.
xwc: Word from standard input at line 2 cut: 'éc...'.
xwc: Word from standard input at line 2 cut: 'gra...'.
xwc: Word from standard input at line 2 cut: 'ré...'.
xwc: Word from standard input at line 2 cut: 'rou...'.
xwc: Word from standard input at line 2 cut: 'tre...'.
xwc: Word from standard input at line 2 cut: 'Tou...'.
xwc: Word from standard input at line 2 cut: 'br♦...'.
Le bâton d'épine pétillait et jetait des étincelles jusqu'au milieu de
xwc: Word from standard input at line 3 cut: 'bâ...'.
xwc: Word from standard input at line 3 cut: 'ép...'.
xwc: Word from standard input at line 3 cut: 'pé...'.
xwc: Word from standard input at line 3 cut: 'jet...'.
xwc: Word from standard input at line 3 cut: 'ét...'.
xwc: Word from standard input at line 3 cut: 'jus...'.
xwc: Word from standard input at line 3 cut: 'mil...'.
la chambre.
xwc: Word from standard input at line 4 cut: 'cha...'.
--- ends reading for #1 FILE
⟶""
au⟶1
Au⟶1
bâ⟶1
bou⟶1
br♦⟶1
```

```
cha⟶2
d⟶3
de⟶2
des⟶1
éc⟶1
en⟶1
ép⟶1
et⟶3
ét⟶1
fac⟶1
fur⟶1
gra⟶1
jet⟶1
jus⟶1
la⟶2
le⟶1
Le⟶1
mil⟶1
mur⟶1
pé⟶1
que⟶1
ré⟶1
rou⟶1
sec⟶1
Tou⟶1
tre⟶1
une⟶1
```

► 19 Dans le projet, 'é' est un mot de longueur 1 ? 'ééé' est le mot mémorisé suite a la lecture du mot 'éééé' avec l présence de l'option –initial=3 ?

Avec `fr_FR.UTF8`, « é » est une chaine de longueur 2. Avec en plus la valeur de `--initial` fixée à 3, « éééé » sera coupé en une chaine de longueur 3, dont les deux premiers *bytes* correspondent à « é ».

► 20 Dans les paramètres de la ligne de commande, comment savoir s'il s'agit d'une option ou d'un nom de fichier ? Il faut essayer d'ouvrir avec `fopen` à chaque fois ?

Par défaut : tout ce qui ne commence pas par « - » ou « -- » correspond (dans le projet) à un nom fichier. Pour toutefois laisser la possibilité d'avoir des noms de fichiers qui commencent par « - » ou « -- », « -- » est donné avant le nom du fichier. Par exemple :

```
$ ./xwc xxx -- yyy -- -zzz -- --help -- -i --punctuation-like-space
```

tente de traiter les fichiers « `xxx` », « `yyy` », « `-zzz` », « `--help` » et « `-i` » avec l'option longue `--punctuation-like-space`.

► 21 Et - et --, ça peut marcher avec -r ? Parce qu'après -r, c'est un nom de fichier qui est attendu.

« `-r-` », « `-r"-"` », « `-r -` », « `-r "-"` », « `--restrict=-` » ou « `--restrict="-"` » pour que le fichier restricteur soit l'entrée standard. Et « `-r"--` *file*`"` », « `-r "--` *file*`"` » ou encore « `-restrict="--` *file*`"` » pour que le fichier de nom *file*, qu'il commence par « - », par « -- » ou par ni l'un ni l'autre, soit le fichier restricteur ; attention toutefois à ne donner qu'une seule espace entre « -- » et *file*.

► 22 Auriez-vous d'autres textes sur lesquels faire des tests que les totos et les misérables ?

Ont été ajoutés : les « Totos » sous la forme de fichiers, sans encodage, `toto`*n*`.txt`, et en UTF8 (avec accents), `toto`*n*`_.txt` ; Tartuffe ou l'Imposteur, sans encodage, `tartuffe.txt`, et en UTF8, `tartuffe_.txt` ; `abeeeillmrsss.txt` et `sssrmllieeeba.txt`, avec tous les mots des Misérables sans encodage issus de `lesmiserables.txt`, dans l'ordre croissant puis décroissant selon `strcoll` dans l'environnement « `LANG=fr_FR.UTF-8` » ; la version UTF8 des Misérables, `lesmiserables_.txt` ; la liste (croissante) des mots du français utilisée par certains outils de correction orthographique, `fr_.txt`.

► 23 Je n'envisage que le traitement des mots à partir d'une table de hachage, et pas d'un arbre binaire donc. Est-ce que je dois gérer les options `-w`, `--words-processing`, `-a` ou `-h` ?

Sauf si c'est déjà fait, ne passez pas du temps à cela. Pensez en revanche à l'indiquer dans l'aide. Après « *Exclusive word counting. Print…* » : « *Process words using a hash table.* », « *Process words using hash tables.* », « *Process words using an AVL binary tree.* » ou « *Process words using AVL binary trees.* ». Et évidemment dans votre rapport.

Si c'est fait, prévoyez un message d'erreur comme « *Undeveloped option* ».

► 24 Est-ce que nos messages d'avertissement et d'erreur doivent être exactement les mêmes que les vôtres ?

Non. Les miens sont donnés à titre indicatif, dont les numéros de ligne des mots coupés puisque que c'est cela qui vous tracasse… Leurs libellés me semblent toutefois justes et conformes à ce qui serait attendu d'un locuteur linux-globish. Passez-les en français si cela vous chante. De même l'aide. Mais en aucun cas les identificateurs des options.

► 25 Je ne comprends pas pourquoi dans vos exemples les flèches n'ont pas la même taille. Et je ne sais pas quels symboles on doit utiliser pour les dessiner.

Les colonnes sont séparées par le caractère de tabulation. Les flèches signifient dans mes exemples l'affichage de ces caractères de tabulation lorsque les largeurs des tabulations sont fixées à 8 ou à 16, autrement dit lorsque des taquets de tabulation sont disposés tous les 8 ou 16 caractères à partir du premier caractère. Vous ne devez donc rien dessiner : juste employer ces caractères. Vous pouvez obtenir un affichage différent en modifiant les taquets de tabulation du terminal *via* la commande `tabs`. Si vous voulez voir les contenus des colonnes bien alignés sans avoir affaire au problème de la largeur de fenêtre, éditez la sortie sous votre tableur favori en la considérant comme un fichier ou format CSV avec la tabulation comme séparateur de colonnes.

► 26 Est-ce que nos sorties doivent être exactement les mêmes que les vôtres ?

Oui, sauf sauf si aucun tri n'est pratiqué sur la sortie. En supposant que l'exécutable fourni figure dans le dossier `~/Téléchargements/`, que votre exécutable figure dans le dossier courant : si aucun tri est utilisé, testez en donnant du

```
$ ~/Téléchargements/xwc options files | sort > tmp1.txt \
&& ./xwc options files | sort > tmp2.txt \
&& diff tmp1.txt tmp2.txt \
&& rm tmp1.txt tmp2.txt
```

et si un tri est utilisé, testez avec

```
$ ~/Téléchargements/xwc options files > tmp1.txt \
&& ./xwc options files > tmp2.txt \
&& diff tmp1.txt tmp2.txt \
&& rm tmp1.txt tmp2.txt
```

Si la commande composée affiche quoique ce soit, il y a un problème.

— *Quoi d'autre ?*