R1.04 – TP 4

Table des matières

1 Codage des caractères

Dans une ressource précédente (R1.03), vous avez découvert le codage des entiers et des flottants. Un troisième type d'information à coder est le caractère. Le codage des caractères recouvre deux notions :

1. un entier appelé point de code qui est le numéro du caractère dans la liste des caractères à coder,
2. la représentation en binaire de ce point de code appelée codet

Pour les codes les plus simples, la manière de représenter le point de code en binaire est unique, et on confond les deux notions comme pour l'ASCII. Pour unicode, le codage de caractères le plus utilisé actuellement, il existe plusieurs représentation binaire possibles du point de code : ucs-2, utf-8, ...

Le codage des caractères n'intègre pas la représentation graphique ou glyphe de ce caractère.

ASCII (*American Standard Code for Information Interchange*) est un codage de caractères sur 7 bits (il existe donc 128 codes). ASCII intègre les chiffres, les lettres de l'alphabet latin (majuscules et minuscules), des caractères de ponctuation. Par contre, ASCII n'inclut pas de caractères accentués. Pour les intégrer, une déclinaison sur 8 bits (donc comportant 256 caractères) a été définie. Mais, les besoins des différentes langues n'étant pas uniformes, il en existe plusieurs version. ISO_8859-1 (ou Latin-1) a été l'une des plus populaires. Ces différentes extensions sont incompatibles entre elles.

Unicode (ISO 10646) est le standard actuel et vise à représenter tous les caractères de chacune des langues humaines (et même plus !). Unicode compte actuellement près de 150 000 caractère : `https://www.unicode.org/charts/`. Unicode différencie les deux notions du codage en établissant une liste des caractères avec leur point de code et plusieurs manières de coder ces points de code. Parmi les codages disponible :

1. UCS-2, UCS-4 (*Universal Character Set*) : permet de représenter les caractères par le codage binaire du point de code sur 2 ou 4 octets.
2. UTF-8 (*Universal Character Set Transformation Format*) : codage sur 1 à 4 octets avec la particularité que ASCII et UTF-8 coïncident sur les 128 caractères : `https://fr.wikipedia.org/wiki/UTF-8#Description`.

Question.

— Consultez la valeur de la variable d'environnement `LANG`. Déduisez-en le codage de caractères que vous utilisez.
— Consultez le manuel de `xxd` qui permet d'obtenir un dump hexadécimal de son entrée. Vous regarderez en particulier les options `-r` et `-p`.
— Saisissez `echo -n 'e' |xxd -p` et, à l'aide du man d'ascii, vérifiez que ASCII et UTF-8 coïncident.
— Consultez le man de `iconv`, en particulier les options `-f -t` et `-l`.
— Trouvez le codage de `'e'` en UCS-2 et UCS-4
— Quelle est la représentation de `'é'` en UTF-8 ?
— Quelle est la représentation de `'ç'` en ISO_8859-1. Vérifiez avec `man iso_8859-1`.
— Quelle est la représentation de `'è'` en UCS-2 ?
— Quelles sont les représentations de `'€'` et `'→'` en UTF-8, UCS-2 et UCS-4 ?
— Même question pour le caractère `1F042` (à copier-coller depuis `https://www.unicode.org/charts/`). Cela explique-t-il pourquoi UCS-2 est maintenant considéré comme obsolète ?
— À quels caractères correspond cette séquence UTF-8 : `44c3a96ac3a0` ?
— À quels caractères correspond cette séquence UTF-8 : `f09d849e20cea620e289a4` ?
— À quels caractères correspond cette séquence ISO_8859-1 : `f4d7303dd8` ?
— À quels caractères correspond cette séquence UCS-4 : `00002615000003a90001f0b7` ?

2 Recherche

2.1 Recherche de fichier

La recherche de fichiers se fait par la commande `find`. Cette commande comporte essentiellement trois clauses :

1. où chercher,
2. des critères de sélection, comme `-iname`, `-type`, `-size`, ...
3. une action : par défaut `-print` = affichage, mais on peut également utiliser `ls -exec` ou `-execdir` qui permettent

Question.

— consulter le manuel de `find`
— Copiez le fichier `gen.sh` que vous trouverez sur Madoc dans votre répertoire courant. Il s'agit d'un *script* qui exécute une suite de commandes afin de générer aléatoirement une arborescence de fichiers dans votre répertoire courant.
— Rendez `gen.sh` exécutable pour vous. Exécutez le par la commande `./gen.sh 100 400 10 arbo` afin de créer une arborescence de fichiers située dans le dossier `arbo` (de votre répertoire courant) contenant 100 dossiers et 400 fichiers avec une profondeur de 10 au maximum.

— Après vous être placé dans le dossier `arbo`, énumérez tous les fichiers et dossiers contenus dans ce dossier.
— Énumérez tous les fichiers portant le nom `athing.java` contenus dans `arbo`.
— Énumérez tous les `mp3` contenus dans `arbo`.
— Énumérez tous les fichiers qui ne portent pas l'extension `.txt` dans `arbo` (attention, il ne faut pas énumérer de dossiers).
— Donnez les fichier `.jpg` de plus de 1k, triés par taille
— Supprimez les fichiers `.java`
— Combien y a-t-il de noms de fichier `.txt` différents ?

2.2 Recherche de motif dans un fichier

`find` permet de chercher un fichier mais par un motif dans celui-ci. Cette recherche de motif se fait à l'aide de la commande `grep`. `grep` affiche les lignes ou noms des fichiers où l'on trouve le motif recherché. **Question.**

— Consultez le manuel de `grep`. Vous étudierez en particulier les options `-i` et `-v`.
— Reprenez le fichier `personnes.csv` du Tp1 et affichez les lignes comportant `Gabrielle`.
— Comptez le nombre de personne de sexe féminin (en utilisant le champs titre).
— Triez les lignes comportant `Gabrielle` par nom de famille.
— Comptez le nombre de personnes prénommées `Martin`. Pas les personnes nommées `Martin`, ni `Martinez`, ...
— Donnez (le numéro de ligne où apparait la première fois le prénom `Emilie`.