

Environnement de développement (2I012)

Partiel – Durée : 2h

Valérie Ménissier-Morain & Christoph Lauter

23 mars 2016

Conditions générales

Documents autorisés

Tous les documents sont autorisés. L'usage des clefs USB en lecture seule est autorisé.

Le sujet est disponible depuis la page Web du module <https://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2015/ue/2I012-2016fev/>.

La version HTML directement disponible sur la page Web est particulièrement avantageuse puisque les lignes très longues y peuvent être mieux vues et les caractères UTF-8 non ASCII y peuvent être copiés correctement pour être collés dans un terminal.

Traitement général du sujet

Parmi les utilitaires Unix qui peuvent être employés à bon escient dans cet examen, citons `sort`, `uniq`, `cut`, `head`, `tail`, `grep`, `sed`, `mktemp` (et bien sûr `bash`). **La commande `tr` pourra être utilisée notamment pour remplacer un délimiteur par un retour à la ligne `\n`.**

Comme l'indique le barème, certaines questions sont difficiles. Pensez d'abord à traiter les questions faciles puis correctement les cas simples des questions difficiles plutôt que de vous acharner à traiter tous les cas particuliers.

Il est de loin préférable d'avoir moins de scripts mais qui marchent, plutôt que d'avoir essayé de toucher à tout et de n'avoir rien qui marche à la fin.

N'oubliez pas de tester vos scripts ! Vous avez des fichiers d'exemples à votre disposition, vous avez des exemples textuels dans le sujet que vous pouvez copier/coller, et vous pouvez si ça ne suffit pas créer vos propres exemples de toutes pièces.

Le barème est indicatif et correspond à un total de 29. La note finale est la somme des points à chacune des questions tronquée à 20. Par exemple, une note de 18/29 est ramenée à 18/20 et une note de 22/29 est ramenée à 20/20.

Fichiers à rendre

Tous les fichiers que vous aurez à écrire doivent être placés dans un répertoire nommé `envdev` placé à la racine de votre répertoire personnel.

Le contenu du répertoire `envdev` (et seulement celui-là) sera récupéré par l'équipe système de l'ARI dans vos `HOME` à l'issue de l'épreuve.

Les scripts que vous aurez à écrire ne doivent pas dépendre du répertoire où ils sont placés ; ils ne doivent pas contenir de référence à des chemins absolus et doivent pouvoir être exécutés sans problème depuis n'importe quel répertoire. Pour vos tests vous exécuterez au préalable

```
PATH=~ /envdev : $PATH
```

Un script s'édite, sous Emacs, en mode *shell-script*, ce que l'on peut obtenir en forçant le mode avec `C-c b` si vous avez créé ce raccourci ou `M-x sh-mode` sinon. Dans ce mode la commande `C-c :` place la convention du *shebang* au début de votre script et le rend exécutable.

Même si vous n'utilisez pas Emacs, n'oubliez pas de rendre vos scripts exécutables par `chmod +x script` et en mettant un *shebang* fonctionnel au début du fichier.

De plus, quand il vous est demandé un script *script*, vous devez rendre le script *script* et non un script *script.sh*.

Fichiers de données

Vous trouverez un certain nombre de fichiers de données dans le répertoire [/Infos/lmd/2015/licence/ue/2I012-2016fev/Partiel](https://www-licence.ufr-info-p6.jussieu.fr/lmd/2015/licence/ue/2I012-2016fev/Partiel).

Vous n'avez pas besoin de modifier ces fichiers de données, alors ne les copiez pas ! Déclarez plutôt dans votre terminal de test une variable `DONNEES` par

```
DONNEES=/Infos/lmd/2015/licence/ue/2I012-2016fev/Partiel
```

et utilisez ensuite `$DONNEES/fichier`.

Nous aurons recours à ces différents fichiers dans les exemples ci-dessous.

Exécution et tests

Vos scripts doivent être **testés**. Pour cela nous vous conseillons fortement de copier-coller les exemples depuis le sujet au format HTML présent sur le site Web de l'UE.

CES TRACES D'EXÉCUTION SONT CONTRACTUELLES : SUR LE MÊME JEU DE DONNÉES VOTRE SCRIPT DOIT RENDRE EXACTEMENT CE RÉSULTAT !

Aucun de vos scripts ne doit laisser après son exécution de fichier non prévu et tout fichier temporaire devra être créé avec la commande `mktemp`, par exemple par

```
fichier_temporaire=`mktemp`
```

et supprimé à la fin de vos scripts par


```
rm -f $fichier_temporaire
```

Mais la plupart du temps l'utilisation d'un fichier temporaire n'est qu'une solution de facilité qui masque une mauvaise maîtrise de la composition de commandes.

Vous attacherez un soin particulier au **test** de vos programmes et il est beaucoup, beaucoup, beaucoup, **beaucoup, beaucoup** plus important d'avoir quelques programmes qui fonctionnent qu'avoir touché à tout sans que rien ne fonctionne.

Conventions typographiques

Nous utilisons dans ce sujet la convention typographique utilisée dans le cours pour les exemples : les lignes en gras commençant par % correspondent à des commandes, le % ne fait pas partie de la commande (c'est le prompt du shell dans notre convention), les lignes qui suivent (jusqu'à une nouvelle ligne en gras ou la fin du paragraphe) sont le résultat de l'exécution de la commande.

Par ailleurs ce sujet comporte des lignes dont la longueur excède la largeur d'une feuille de papier. Dans la version papier les longues lignes sont typographiées en plusieurs lignes, une petite main  indique que la ligne n'est pas terminée et continue sur celle qui suit. La version HTML ne présente pas la même contrainte de longueur de lignes et si ces coupures artificielles vous troublent ou si vous avez besoin de faire des copier-coller vous vous y reporterez avec profit.

Remarques générales

Si votre script ne fonctionne pas, pensez à l'option `-x` de `bash` qui permet de tracer l'exécution du script (préfixez votre script par `bash -x`) et si vous souhaitez visualiser cet affichage verbeux en page par page alors

```
bash -x script arguments 2>&1 | less
```

Lire les consignes et conseils

(feuille à part distribuée avant le début de l'épreuve)

et le sujet entièrement n'est pas une perte de temps !

Il est impératif de tester vos scripts : le sujet est disponible depuis la page Web du module, pensez-y pour *copier-coller* les tests de l'énoncé.

Trains allemands

Le fichier `$DONNEES/lines23.csv` contient la description de tous les trains grandes lignes voyageant en Allemagne aujourd'hui, à raison d'un train par ligne du fichier. Un train est décrit par une ligne de la forme suivante (délimiteur de champs `|`) :

avec :

- IDENT son identifiant sous forme d'une chaîne de caractères (lettres, chiffres et espace), tel que Bus 134 ou CNL40470,
- ARRET les arrêts successifs (en nombre variable).

Chaque arrêt est de la forme suivante (sous-délimiteur de champs ;) :

NOM;NUMERO;ARRIVEE;DEPART;VOIE

avec :

- NOM le nom de la gare, une chaîne de caractères au format UTF-8 (celui de ce document, celui du fichier et par défaut celui de votre terminal),
- NUMERO le numéro de la gare sur cette ligne (les arrêts sont numérotés dans l'ordre de parcours à partir de 0),
- ARRIVEE l'horaire d'arrivée du train en gare ou un tiret – si la station est une gare de formation du train, item DEPART l'horaire de départ du train en gare ou un tiret – si la station est un terminus,
- VOIE le nom de la voie où le train sera dans cette gare, sous forme de chaîne de caractères (souvent un nombre ou une lettre mais cela peut être plus varié).

Un horaire est de la forme *aaaa-mm-jj hh:mm* c'est-à-dire le jour sous la forme année sur 4 chiffres, mois et jour sur 2 chiffres séparés par des tirets –, puis l'heure sous la forme heure et minutes sur 2 chiffres séparés par :. Par exemple cet examen commence selon ce format à 2016-03-23 10:45. On appellera dans la suite ce format un *horaire de train*.

Attention, les différentes informations contenues dans une ligne peuvent presque toutes contenir des espaces.

Voici le début du fichier :

```
% head -n3 $DONNEES/lines23.csv
IC 2443|Köln Hbf;0;-;2016-03-23 07:13;2|Solingen Hbf;1;2016-03-23 07:29;
2016-03-23 07:31;3|Wuppertal Hbf;2;2016-03-23 07:41;2016-03-23 07:43;
2|Hagen Hbf;3;2016-03-23 07:59;2016-03-23 08:01;6|Dortmund Hbf;4;
2016-03-23 08:21;2016-03-23 08:28;10|Hamm (Westf);5;2016-03-23 08:42;
2016-03-23 08:44;9|Gütersloh Hbf;6;2016-03-23 09:05;2016-03-23 09:07;
3|Bielefeld Hbf;7;2016-03-23 09:17;2016-03-23 09:19;2|Herford;8;
2016-03-23 09:26;2016-03-23 09:28;4|Bad Oeynhausen;9;2016-03-23 09:37;
2016-03-23 09:39;1|Minden (Westf);10;2016-03-23 09:47;2016-03-23 09:49;
13|Hannover Hbf;11;2016-03-23 10:18;2016-03-23 10:36;10|Braunschweig Hbf;
12;2016-03-23 11:09;2016-03-23 11:11;7|Magdeburg Hbf;13;2016-03-23 11:56;
2016-03-23 12:01;5|Köthen;14;2016-03-23 12:28;2016-03-23 12:30;
4|Halle (Saale) Hbf;15;2016-03-23 12:52;2016-03-23 12:54;1|Leipzig Hbf;16;
2016-03-23 13:17;2016-03-23 13:29;13|Riesa;17;2016-03-23 13:59;2016-03-23 14:01;
2|Dresden-Neustadt;18;2016-03-23 14:30;2016-03-23 14:32;7|Dresden Hbf;19;
2016-03-23 14:37;-;1
Bus 130|Luxembourg;0;-;2016-03-23 13:15;-|Saarbrücken Hbf;1;2016-03-23 14:35;-;-
ICE 693|Berlin Ostbahnhof;0;-;2016-03-23 15:23;6|Berlin Hbf;1;2016-03-23 15:30;
2016-03-23 15:34;13|Berlin-Spandau;2;2016-03-23 15:46;2016-03-23 15:48;
4|Braunschweig Hbf;3;2016-03-23 16:56;2016-03-23 16:58;6|Hildesheim Hbf;
4;2016-03-23 17:23;2016-03-23 17:25;3|Göttingen;5;2016-03-23 17:53;
2016-03-23 17:55;10|Kassel-Wilhelmshöhe;6;2016-03-23 18:14;2016-03-23 18:16;
2|Fulda;7;2016-03-23 18:46;2016-03-23 18:48;4|Hanau Hbf;8;2016-03-23 19:27;
2016-03-23 19:29;6|Frankfurt (Main) Hbf;9;2016-03-23 19:44;2016-03-23 19:50;
9|Mannheim Hbf;10;2016-03-23 20:27;2016-03-23 20:30;5|Stuttgart Hbf;
11;2016-03-23 21:08;2016-03-23 21:13;16|Ulm Hbf;12;2016-03-23 22:07;
2016-03-23 22:09;2|Augsburg Hbf;13;2016-03-23 22:53;2016-03-23 22:55;
4|München-Pasing;14;2016-03-23 23:20;2016-03-23 23:22;9|München Hbf;15;
2016-03-23 23:29;-;13
```

Vous trouverez dans le répertoire de données deux autres fichiers \$DONNEES/bus.csv et \$DONNEES/extrait.csv. Dans la suite nous appellerons un tel fichier un *fichier de trains*.

1 Trains

Question 1 – liste_trains – 1 point

Écrire un script `liste_trains` qui prend en paramètre un *fichier de trains* et qui affiche la liste des trains classés par ordre lexicographique.

Par exemple

```
% liste_trains $DONNEES/lines23.csv | tail -n3
TGV 9578
TGV 9581
TGV 9583

% liste_trains $DONNEES/bus.csv | head -n3
Bus 113
Bus 115
Bus 116
```

Question 2 – origine_train – 1 point

Écrire un script `origine_train` qui prend en paramètre un identifiant de train et un *fichier de trains* et qui affiche la gare de départ de ce train.

Par exemple

```
% origine_train "ICE 693" $DONNEES/lines23.csv
Berlin Ostbahnhof

% origine_train "ICE 11" $DONNEES/lines23.csv
Bruxelles Midi
```

Question 3 – trains_a_partir_de – 3 points

Écrire un script `trains_a_partir_de` qui prend en paramètre un nom de gare et un *fichier de trains* et qui affiche les identifiants des trains qui partent de cette gare triés par ordre lexicographique.

Par exemple

```
% trains_a_partir_de "Stuttgart Hbf" $DONNEES/lines23.csv | head -n3
IC 1074
IC 1114
IC 181

% trains_a_partir_de "Luxembourg" $DONNEES/bus.csv | head -n3
Bus 116
Bus 118
Bus 122

% trains_a_partir_de "Berlin Hbf" $DONNEES/lines23.csv
EC 41
EC 43
EC 45
EC 47
EC 55
```

2 Lignes

Question 4 – liste_arrets_ligne – 2 points

Écrire un script `liste_arrets_ligne` qui prend en paramètre un identifiant de train et un *fichier de trains* et qui affiche la liste des arrêts successifs de cette ligne.

Par exemple

```
% liste_arrets_ligne "IC 2316" $DONNEES/lines23.csv
Stuttgart Hbf
Wiesloch-Walldorf
Heidelberg Hbf
Mannheim Hbf
Mainz Hbf

% liste_arrets_ligne "ICE 693" $DONNEES/lines23.csv | head -n3
Berlin Ostbahnhof
Berlin Hbf
Berlin-Spandau

% liste_arrets_ligne "ICE 693" $DONNEES/lines23.csv | tail -n6
Mannheim Hbf
Stuttgart Hbf
Ulm Hbf
Augsburg Hbf
München-Pasing
München Hbf

% liste_arrets_ligne "ICE 11" $DONNEES/lines23.csv | wc -l
8
```

Question 5 – liste_gares – 3 points

Écrire un script `liste_gares` qui prend en paramètre un *fichier de trains* et qui affiche la liste de toutes les gares mentionnées dans ce fichier par ordre alphabétique.

Par exemple

```
% liste_gares $DONNEES/bus.csv
Berlin Hbf
Frankfurt (Oder)
Luxembourg
Saarbrücken Hbf

% liste_gares $DONNEES/lines23.csv | head -n5
Aachen Hbf
Aachen Süd(Gr)
Aalen
Aarau
Ainring

% liste_gares $DONNEES/lines23.csv | wc -l
466
```

Question 6 – destination_train – 2 points

Écrire un script `destination_train` qui prend en paramètre un identifiant de train et un *fichier de trains* et qui affiche le nom de la gare terminus (la dernière de la ligne).

Par exemple

```
% destination_train "ICE 693" $DONNEES/lines23.csv
München Hbf

% destination_train "ICE 11" $DONNEES/lines23.csv
Frankfurt (Main) Hbf
```

Question 7 – horaire – 3 points

Écrire un script `horaire` qui prend en paramètre une gare, un identifiant de train et un *fichier de trains* et qui indique l'horaire et la voie de départ de ce train à partir de cette gare. Si la gare est le terminus de ce train le script affiche Terminus.

Par exemple

```
% horaire "Mannheim Hbf" "ICE 693" $DONNEES/lines23.csv
```

```
Départ 2016-03-23 20:30 voie 5
```

```
% horaire "Berlin Ostbahnhof" "ICE 693" $DONNEES/lines23.csv
```

```
Départ 2016-03-23 15:23 voie 6
```

```
% horaire "München Hbf" "ICE 693" $DONNEES/lines23.csv
```

```
Terminus
```

Question 8 – trajet_direct – 5 points

Écrire un script `trajet_direct` qui prend en paramètre deux gares et un *fichier de trains* et qui affiche, pour chaque train circulant de la première à la seconde gare l'identifiant du train, l'heure de départ de la première gare et l'heure d'arrivée à la seconde gare, informations séparées par des virgules. S'il y a plusieurs trains alors ils seront ordonnés selon l'horaire de départ.

Par exemple

```
% trajet_direct "Hannover Hbf" "Karlsruhe Hbf" $DONNEES/lines23.csv
```

```
ICE 103,2016-03-23 05:40,2016-03-23 10:58  
IC 2271,2016-03-23 06:01,2016-03-23 10:50  
IC 2371,2016-03-23 12:01,2016-03-23 16:50  
IC 2375,2016-03-23 16:01,2016-03-23 20:54  
ICE 1171,2016-03-23 17:41,2016-03-23 21:08  
ICE 273,2016-03-23 19:41,2016-03-23 23:08  
CNL 479,2016-03-23 22:01,2016-03-24 04:00  
IC 60479,2016-03-23 22:01,2016-03-24 04:00
```

```
% trajet_direct "Karlsruhe Hbf" "Zürich HB" $DONNEES/lines23.csv
```

```
ICE 3,2016-03-23 05:56,2016-03-23 09:00  
ICE 271,2016-03-23 06:58,2016-03-23 10:00  
EC 9,2016-03-23 13:49,2016-03-23 17:00  
CNL40419,2016-03-24 04:02,2016-03-24 08:05  
CNL 479,2016-03-24 04:02,2016-03-24 08:05  
IC 60479,2016-03-24 04:02,2016-03-24 08:05  
IC 61419,2016-03-24 04:02,2016-03-24 08:05  
CNL 458,2016-03-24 05:09,2016-03-24 09:05  
CNL 471,2016-03-24 05:09,2016-03-24 09:05  
IC 60458,2016-03-24 05:09,2016-03-24 09:05  
IC 60471,2016-03-24 05:09,2016-03-24 09:05
```

3 Dates et horaires

Nous fournissons dans le répertoire `$DONNEES` pour la suite du sujet un script `conversion_date_secondes` qui prend en paramètre un *horaire de train* et qui affiche le nombre de secondes écoulées depuis le 1^{er} janvier 1970 correspondant.

Ce fichier est tout petit, vous pouvez le copier dans votre répertoire de travail.

Par exemple

```
% conversion_date_secondes "2016-03-23 14:00"
```

```
1458738000
```

```
% conversion_date_secondes "2016-03-23 16:00"
```

```
1458745200
```

Question 9 – date_posterieure – 1 point

Écrire un script `date_posterieure` qui prend en paramètre deux *horaires de train* et qui renvoie un code de retour nul si et seulement si le second horaire est postérieur au premier. En cas d'horaires identiques le code de retour sera non nul.

Par exemple

```
% date_posterieure "2016-03-23 14:00" "2016-03-23 16:00" && echo "Après"
Après
```

```
% date_posterieure "2016-03-23 16:00" "2016-03-23 14:00" || echo "Avant"
Avant
```

4 Horaires et trains

Question 10 – `trajet_posterieur` – 4 points

Écrire un script `trajet_posterieur` qui prend en paramètre deux gares, un *horaire de train* et un *fichier de trains*, et qui affiche l'identifiant du prochain train au départ de la première gare à destination de la seconde gare et dont l'horaire de départ de la première gare est postérieur à l'horaire donné en paramètre.

Par exemple

```
% trajet_posterieur "Mannheim Hbf" "München Hbf" "2016-03-23 16:00" $DONNEES/lines23.csv
ICE 599,2016-03-23 16:30,2016-03-23 19:27
ICE 611,2016-03-23 17:31,2016-03-23 20:27
ICE 691,2016-03-23 18:30,2016-03-23 21:27
ICE 613,2016-03-23 19:31,2016-03-23 22:26
ICE 693,2016-03-23 20:30,2016-03-23 23:29
ICE 615,2016-03-23 21:31,2016-03-24 00:27
ICE 619,2016-03-24 01:06,2016-03-24 06:02

% trajet_posterieur "Berlin Ostbahnhof" "München Hbf" "2016-03-23 03:00" $DONNEES/lines23.csv
ICE 1091,2016-03-23 05:53,2016-03-23 13:27
ICE 595,2016-03-23 07:23,2016-03-23 15:28
ICE 597,2016-03-23 09:23,2016-03-23 17:27
ICE 599,2016-03-23 11:23,2016-03-23 19:27
ICE 691,2016-03-23 13:23,2016-03-23 21:27
ICE 693,2016-03-23 15:23,2016-03-23 23:29
```

Question 11 – `correspondance_via` – 4 points

Écrire un script `correspondance_via` qui prend en paramètre trois gares et un *fichier de trains*, et qui affiche tous les trajets qui vont de la première gare à la deuxième en passant par la troisième gare, par ordre de départ, sous la forme de l'identifiant du train suivi. Lorsque plusieurs trains sont possibles pour le second trajet, on ne retiendra que le premier train possible. Dans l'affichage ci-dessous l'espacement en début de lignes est réalisé avec 2 espaces.

Par exemple

```
% correspondance_via "Hannover Hbf" "Zürich HB" "Karlsruhe Hbf" $DONNEES/lines23.csv
Train ICE 103
  Hannover Hbf départ 2016-03-23 05:40 - Karlsruhe Hbf arrivée 2016-03-23 10:58
puis train EC 9
  Karlsruhe Hbf départ 2016-03-23 13:49 - Zürich HB arrivée 2016-03-23 17:00
Train IC 2271
  Hannover Hbf départ 2016-03-23 06:01 - Karlsruhe Hbf arrivée 2016-03-23 10:50
puis train EC 9
  Karlsruhe Hbf départ 2016-03-23 13:49 - Zürich HB arrivée 2016-03-23 17:00
Train IC 2371
  Hannover Hbf départ 2016-03-23 12:01 - Karlsruhe Hbf arrivée 2016-03-23 16:50
puis train CNL40419
  Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05
Train IC 2375
  Hannover Hbf départ 2016-03-23 16:01 - Karlsruhe Hbf arrivée 2016-03-23 20:54
puis train CNL40419
  Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05
Train ICE 1171
  Hannover Hbf départ 2016-03-23 17:41 - Karlsruhe Hbf arrivée 2016-03-23 21:08
puis train CNL40419
```

Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05
Train ICE 273
Hannover Hbf départ 2016-03-23 19:41 - Karlsruhe Hbf arrivée 2016-03-23 23:08
puis train CNL40419
Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05
Train CNL 479
Hannover Hbf départ 2016-03-23 22:01 - Karlsruhe Hbf arrivée 2016-03-24 04:00
puis train CNL40419
Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05
Train IC 60479
Hannover Hbf départ 2016-03-23 22:01 - Karlsruhe Hbf arrivée 2016-03-24 04:00
puis train CNL40419
Karlsruhe Hbf départ 2016-03-24 04:02 - Zürich HB arrivée 2016-03-24 08:05