ArcSys1 : Systèmes et Réseaux

Examen final du 13 décembre 2021 (2 h)

Tous documents interdits à l'exception d'un A4 recto manuscrit de votre main et du pense-bête de C fourni en cours. Calculatrices, téléphones, et autres objets connectés interdits. La correction tiendra compte de la qualité de l'argumentaire et de la présentation. Un code illisible ou incompréhensible est un code faux. 1 point de barème ≈ 6 minutes.
Le sujet est divisé en deux parties à rédiger sur des copies séparées.

Partie 1 : Programmation en C (1h 12 min – 12 points).

À rendre sur une copie séparée.

★ Exercice 1 : Javanais (30 min – 5 pts).

Le javanais est un jeu sur la langue française comparable au verlan. On change les mots pour les rendre difficilement compréhensibles, sauf pour les habitués. Le principe de base est d'insérer les lettres 'av' avant chaque groupe de voyelles. Par exemple, "un bon abricot" devient "avun bavon avabravicavot".

▷ **Question 1 :** *Javanais simplifié.*

On souhaite écrire une fonction écrivant dans le buffer fourni la version javanaise du mot passé en premier argument. Son prototype est le suivant :

```
void javanais(char* mot, char* buffer);
```

On supposera que le buffer est toujours assez grand pour stocker le mot en javanais. Dans cette première version, on ajoutera la chaine "av" avant chaque voyelle. Le mot "jour" sera donc traduit en "javoavur", même si la traduction correcte devrait être "javour".

▷ **Question 2 :** *Javanais soutenu.*

Proposez une autre implémentation de la fonction `javanais` permettant de n'insérer "av" qu'une seule fois pour un groupe de voyelles consécutives. Cette fois-ci, "jour" sera bien traduit en "javour" tandis que "ouate" sera traduit "avouatave" et "eau" sera traduit "aveau"

▷ **Question 3 :** *Javanais littéraire.*

Proposez une troisième implémentation de la fonction pour que le "e" muet d'un mot ne soit pas étendu. On souhaite donc avoir les traductions suivantes "bicrave" ⇝ "bavicravave" (et non "bavicravavave"). "poucave" ⇝ "pavoucavave" et non "pavoucavavave".

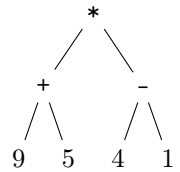
▷ **Question 4 :** *Traducteur complet.*

Écrivez un programme complet prenant le nom d'un fichier en paramètre de la ligne de commande, et affichant la traduction en javanais de son contenu. Il n'est évidement pas nécessaire de recopier la fonction `javanais()` implémentée aux questions précédentes.

Illustration : extraits de la Javanaise de Gainsbourg en javanais littéraire.

J'avoue j'en ai bavé pas vous, mon amour	⇝	J'avavavoue j'aven avai bavavavé pavas vavous, mavon avamavour
Avant d'avoir eu vent de vous, mon amour	⇝	avAvavant d'avavavoir aveu vavent dave vavous, mavon avamavour
A votre avis qu'avons-nous vu de l'amour ?	⇝	avA vavotrave avavavis qavu'avavavons-navous vavu de l'avamavour ?
De vous a moi vous m'avez eu, mon amour	⇝	Dave vavous ava mavoi vavous m'avavavez aveu, mavon avamavour
Helas avril en vain me voue à l'amour	⇝	Havelavas avavravil aven vavain mave vavoue à l'avamavour
J'avais envie de voir en vous cet amour	⇝	J'avavavais avenvavie dave vavoir aven vavous cavet avamavour

★ Exercice 2 : Programmation C (42 min – 7 pts).



Arbre A

On souhaite pouvoir manipuler des expressions arithmétiques sous la forme d'arbres binaires dont les feuilles sont des valeurs, et les nœuds sont les opérateurs arithmétiques `+`, `-`, `*` et `/`. L'arbre A ci-contre représente l'expression $((9+5)*(4-1))$.

```
struct tree {
  unsigned int value;
  struct tree* left;
  struct tree* right;
};
```

On utilisera la structure de données `struct tree` ci-dessus. Les champs `left` et `right` sont respectivement des pointeurs vers les éventuels enfants gauche et droite. On représente l'absence d'enfant par un pointeur `NULL`. Le champ `value` a deux interprétations : dans le cas d'une feuille, il s'agit d'un entier, et dans le cas d'un nœud interne, il s'agit d'un opérateur arithmétique, représenté par l'une des valeurs ASCII suivantes : `'+'`, `'-'`, `'*'` ou `'/'`.

▷ **Question 1 :** Implémentez la fonction `tree_is_leaf` dont le prototype est donné ci-dessous. Elle renvoie `1` si l'arbre `t` est réduit à une feuille et `0` sinon.

```
int tree_is_leaf(struct tree* t);
```

▷ **Question 2 :** Implémentez les fonctions `F` et `N` permettant respectivement d'allouer une feuille dont la valeur est donnée, et un nœud dont les enfants et l'opérateur sont donnés. Donner ensuite le code utilisant ces fonctions permettant d'initialiser l'arbre A.

```
struct tree* F(unsigned int value);
struct tree* N(char op, struct tree* left, struct tree* right);
```

▷ **Question 3 :** Implémentez la fonction `tree_free` permettant de libérer toute la mémoire allouée par un arbre.

```
void tree_free(struct tree* t)
```

▷ **Question 4 :** Implémentez la fonction `tree_eval` permettant de calculer la valeur vers laquelle l'expression associée à un arbre s'évalue. Par exemple, `tree_eval(`A`)` doit renvoyer 42.

```
int tree_eval(struct tree* t);
```

▷ **Question 5 :** On souhaite maintenant pouvoir représenter des arbres d'expression d'arité quelconque, c'est-à-dire dont les nœuds peuvent avoir un nombre arbitraire d'enfants. Implémentez une structure `struct ntree` permettant de représenter ces objets.

▷ **Question 6 :** Implémentez les fonctions `ntree_alloc` (qui alloue une feuille de la valeur donnée), `ntree_add_child` (qui ajoute un enfant à un nœud parent), et `ntree_free` (qui libère la mémoire allouée par un arbre).

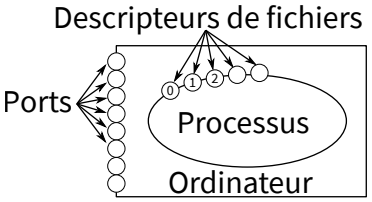
```
1 struct ntree* ntree_alloc(unsigned int value);
2 void ntree_add_child(struct ntree* parent, struct ntree* child);
3 void ntree_free(struct ntree* t);
```

▷ **Question 7 :** Implémentez une fonction `ntree_parse`, qui permet de créer un arbre à partir d'une chaîne de caractères. Par exemple, à partir de la chaîne de caractères `"((9+5)*(4-1))"`, on veut générer l'arbre A. Votre implémentation peut supposer que les chaînes suivent scrupuleusement le formalisme attendu, sans gérer le cas des chaînes mal formées.

Partie 2 : Réseau (48 min – 8 points). À rendre sur une copie séparée.

★ Exercice 3 : Sockets BSD (12 min – 2 pts).

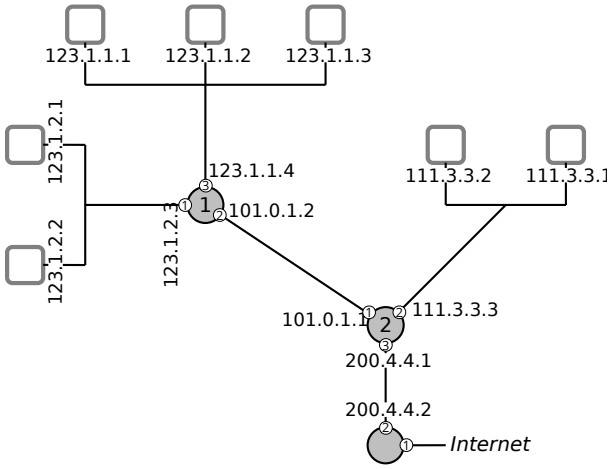
On rappelle le formalisme vu en cours pour représenter des processus en cours d'exécution, repris ci-contre. Les machines sont représentées par des rectangles, avec des petits ronds sur le coté pour représenter les ports de la machine. Au sein des machines, chaque processus est représenté par une ellipse, et des petits ronds à la bordure de cette ellipse représente les différents descripteurs de fichier du processus. Les trois premiers sont habituellement `stdin`, `stdout`, `stderr`.



▷ **Question 1 :** Dessinez en respectant ce formalisme les différentes étapes de l'établissement d'une connexion TCP entre deux processus distants. Vous donnerez les appels systèmes utilisés par chaque partie. Vous dessinerez l'état des choses après chaque appel système, en ajoutant les explications adéquates. En revanche, il ne vous est pas demandé de donner le code exact réalisant ces appels systèmes.

★ Exercice 4 : Routage (12 min – 2pts).

▷ **Question 1 :** Donnez une table de routage pour les routeurs 1 et 2 dans le réseau ci-contre.



★ Exercice 5 : Web et TCP (24 min – 4 pts).

Un navigateur Web fait une requête HTTP auprès d'un serveur pour télécharger une page de $10\,ko$. Le navigateur et le serveur sont localisés sur des machines différentes d'un même réseau Ethernet dont le MTU de 1000 octets [1].

▷ **Question 1 :** Listez tous les datagrammes IP [2] échangés entre le navigateur et le serveur, en explicitant le contenu de chacun en quelques mots.

Le réseau utilisé a une latence nulle ($0\,s$) et une bande passante de $1\,ko/s$. Ces valeurs ne sont pas réalistes, mais elles simplifient les calculs.

▷ **Question 2 :** Calculez le temps nécessaire pour télécharger l'image en supposant qu'il n'y a aucune erreur, aucune perte de paquet et aucune duplication de paquet. On suppose de plus que la connexion TCP reste en mode SS pendant tout l'échange, avec une taille initiale de fenêtre de congestion à 1. Un schéma explicatif semble nécessaire.

1. MTU : Maximum Transmission Unit, taille maximale des paquets transitant sur ce réseau.
2. Datagramme : c'est le nom correct des paquets échangés à la couche IP.