TD/TP 7 : allocation dynamique de mémoire

Gilles Peskine

22 novembre 2002

1 Fonctions de base

(a) Écrire une fonction qui a le même prototype que `malloc`, qui appelle `malloc` et qui arrête le programme en signalant à l'utilisateur un manque de mémoire si `malloc` échoue. Un nom classique pour une telle fonction est `xmalloc` ; on définit de même `xrealloc`.

(b) Écrire une fonction `dupstr` qui fait une copie d'une chaîne de caractères vers un espace mémoire alloué dynamiquement.

```
char *dupstr(char *s);
```

(c) Écrire une fonction `getline` qui lit en mémoire une ligne d'un fichier texte et la stocke dans un espace mémoire alloué dynamiquement.

```
char *getline(FILE *f)
```

2 Dernières lignes d'un fichier : la commande `tail`

La commande Unix `tail` lit un fichier sur son entrée standard et affiche les N dernières lignes sur sa sortie standard, le nombre N étant spécifié sur la ligne de commande précédé du signe « - ». Par exemple, `tail -1` affiche la dernière ligne. Écrire `tail` à l'aide de `strtol` et `getline`.

Rappel :

```
long strtol(char *nptr, char **endptr, int base);
```

3 Recherches d'une sous-chaîne

Écrire une fonction qui recherche une chaîne `mot` dans une autre `s`. Renvoyer un tableau contenant la liste de toutes les positions (pointeurs sur un caractère de `s`), suivies d'un `NULL` pour marquer la fin du tableau.

```
char **recherche_sous_chaines(char *s, char *mot);
```

Exemple :

`recherche_sous_chaine("coucou hou", "ou")` $\Longrightarrow$ `{s + 1, s + 4, s + 8, NULL}`

4 Lecture du fichier `/etc/services`

La plupart des machines sous Unix ont un fichier `/etc/services` qui décrit la correspondance entre les noms et les numéros de port des différents services réseau.

(a) Voici un exemple d'extrait de fichier `/etc/services`.

```
# Network services, Internet style
tcpmux          1/tcp           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
www             80/tcp          http    # WorldWideWeb HTTP
www             80/udp                  # HyperText Transfer Protocol
# Appletalk
echo            4/ddp                   # AppleTalk Echo Protocol
```

Chaque ligne est constituée de deux ou plus de champs séparés par des espaces. Le premier champ est le nom du service (par exemple `echo`, `www`). Le deuxième champ est constitué de deux parties : le numéro de port (par exemple `1`, `80`) et le nom du protocole (par exemple `tcp`, `udp`)[1], séparés par un « `/` ». Les autres champs éventuels (par exemple `http`) sont des noms alternatifs pour le port. Enfin, le caractère « `#` » débute un commentaire qui va jusqu'à la fin de la ligne.

Les données lues de `/etc/services` peuvent être stockées à l'aide du type suivant :

```
struct servent {
    char *s_name;       /* Nom du service */
    char **s_aliases;   /* Pointeur vers un tableau de pointeurs vers les noms
                           alternatifs du service, terminé par un pointeur nul */
    int s_port;         /* Numéro de port */
    char *s_proto;      /* Nom du protocole */
}
```

Écrire une fonction qui lit le fichier `/etc/services` et qui crée et remplit un tableau de `struct servent` avec les informations lues.

(b) Écrire une fonction qui parcourt un tableau de `struct servent` à la recherche d'un service de nom donné et renvoie le numéro de port correspondant. Si le nom de service est invalide, renvoyer `-1`.

```
int numero_de_service(struct servent *db, char *protocole, char *nom);
```

5 Retour sur les exercices précédents

Reprendre les exercices précédents où l'on s'était limité à des tableaux de taille connue à la compilation, et les modifier pour utiliser `malloc`.

[1] les noms et les numéros de ports peuvent être réutilisés dans des protocoles différents