Chaînes de caractères et Fichiers

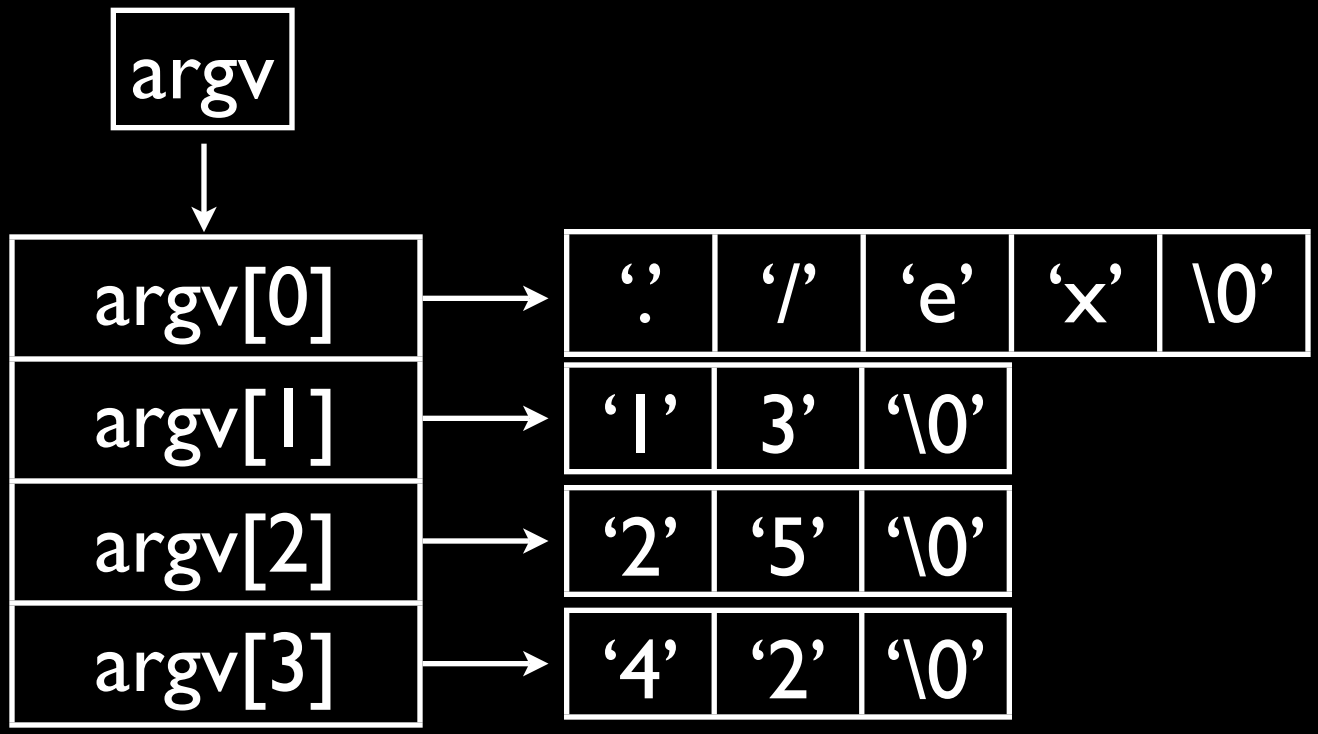
Pierre-Alain FOUQUE

Chaîne de caractères

- un tableau contenant des caractères et finissant par le caractère '\0': char chaine[10];
- chaine[0]='h'; chaine[1]='e'; chaine[2]='l'; chaine[3]='l'; chaine[4]='o'; chaine[5]='\0';
- char chaine[10]="hello"; '\0' automatiquement mis
- char chaine[]; +/- équivalent char *chaine;
- char chaine[]="hello"; crée un tableau de 6 cases
- char *chaine="hello"; Attention: chaine[0]='H'; ne fonctionne pas. Pas de réservation de mémoire
- Allouer soi-même la mémoire

Arguments argv[]

- Le tableau argv[] est de type: char *
- chaque case argv[i] contient un pointeur sur un caractère, ce caractère est le premier de la chaîne i+1

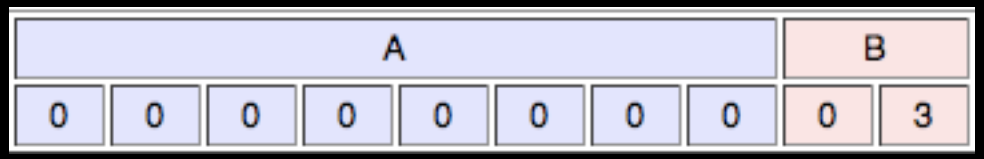


Chaîne de caractères

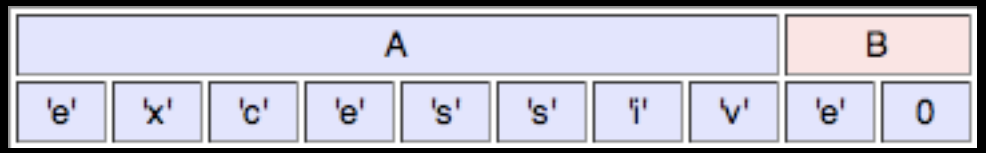
- printf("%s",chaine); pour afficher une chaîne de caractères
- int strlen() donne la longueur
- int strcmp compare dans l'ordre alphabétique
- strncpy recopie n caractères
- atoi : transforme une chaîne en entier
- cf. string.h

strcpy(char *dst, char *src)

- strcpy() ne contrôle pas la taille de la chaîne qui commence à l'adresse src jusqu'au caractère '\0' à traiter et va écrire dans la zone qui commence à l'adresse dst
- On peut écraser les données si taille de la chaîne src > taille de la zone dst allouée



Ecriture de la chaîne "excessive" dans la zone A



Fonction strncpy

- strncpy prend comme argument des adresses
- char *strncpy(char *dst, char *src, int taille);
- char ch1[10]=”coucou”;
- char ch2[10];
- Si on veut mettre dans ch2 les 3 derniers caractères de ch1: strcpy(ch2, ch1+3, 3);

Autres fonctions

- char *strcat(char *s, char *a);
- char *strncat(char *s, char *a, int lg);
- char s[]=”hello ”; char a[]=” world”;
- strcat(s,a) => “helloworld”
- strncat(s,a,2) => “hellowo”
- strstr(chaine,souschaine): recherche première occurrence de souschaine dans chaine

strcat

- `char *strcat(char *s, char *t){`
  - `int i,j;`
  - `i=j=0;`
  - `char *st=malloc(sizeof(char)*(strlen(s)+strlen(t)+1);`
  - `while(s[i] != '\0') {st[i]=s[i]; i++;}`
  - `while((st[i++]=t[j++]) != '\0') ;`
  - `st[i]='\0';`
  - `return st;`
- `}`

atoi

```
int atoi(char *s){
    int i,n=0;
    for (i=0; (s[i]>='0') && (s[i]<='9'); i++)
        n=10*n + (s[i]-'0');
    return n;
}
```

atoi plus générale

```
int atoi(char *s){
    int i,n=0, signe=1;
    for (i=0; s[i]==' ' || s[i]=='\n' || s[i]=='\t'; i++) ;
    if (s[i]=='+') signe=1;
    else if (s[i]=='-') signe=-1;
    for (; (s[i]>='0') && (s[i]<='9'); i++)
        n=10*n + (s[i]-'0');
    return (signe*n);
}
```

Fichiers

- Fichier = pointeur sur une structure « complexe » de type **`FILE`** (**`FILE *`**)
- Ouverture d'un fichier en écriture, lecture

```
FILE *f = fopen(« toto.txt », «rw»);
```

Penser à vérifier si l'opération s'est bien passée

```
if(f==NULL) {
    printf(« erreur dans fopen\n »);
    exit(EXIT_FAILURE);
}
```

- Fermer avec **`fclose(f);`**

3 fichiers spéciaux

- Au lancement du programme, il y a ouverture de 3 fichiers spéciaux: stdin, stdout, stderr
- FILE *stdin est un fichier en lecture, quand on écrit sur le clavier, les données sont stockées dans stdin et on peut le lire: cf. scanf par ex
- FILE *stdout est un fichier en écriture, quand on veut écrire sur l'écran, on écrit dans ce fichier
- FILE *stderr: écrit les erreurs dans stdout

Ecrire dans un fichier

- `fprintf(f, “Coucou %d\n”, a);`
  - comme printf: le premier argument est de type FILE *f;
- `printf(“Hello\n”); fprintf(stdin,“Hello\n”);`
- `int fputc(int c, FILE *f);` caractères à caractères
- `int fputs(char *str, FILE *f);` écrire toute une chaîne
- Ces fonctions déplacent un ptr dans le fichier

Lire dans un fichier

- `char c=fgetc(FILE *f);` caractère à caractère
- `char *fgets(char * str, int size, FILE *f)`
- lit au plus `size-1` caractères et les stockent dans `str` s'arrête dès qu'un caractère `'\n'` est rencontré s'il y en a et ajoute `'\0'` à la fin
- Ces fonctions avancent un pointeur dans le fichier

Fin de fichier

- End Of File: while()
- On peut utiliser la fonction int feof(FILE *f) indique si on est en fin de fichier (si on est en fin de fichier valeur retournée non nulle)
- while((c=fgetc())!=EOF){...}
- Ceci permet de lire le fichier caractère à caractère tant qu'on n'est pas en fin de fichier et de faire un traitement dans les {...}

Conclusion

- Le langage C n'est pas fait pour gérer des chaînes de caractères (possibles, mais plus laborieux que dans un autre langage)