Programmation Procédurale – Entrées / Sorties

Polytech'Nice Sophia Antipolis

Erick Gallesio

2015 – 2016

E/S avec format (1 / 3)

printf

```
int printf(const char *format, ...);
```

- le format spécifie le nombre, le type et les contraintes sur la représentation textuelle de des expressions passées dans la liste variable.
- Exemple:

```
printf("Name: %-15s average: %6.2f rank: %04d", name, avg, rk);
```

scanf

```
int scanf(const char *format, ...);
```

- le format spécifie le nombre, le type et les contraintes sur la représentation textuelle de des objets sont les adresses sont passées dans la liste variable.
- la valeur de retour est le nombre d'items effectivement lus
- Exemple:

```
scanf("%s %*s %f %d", name, &avg, &rk); /* '%*s' = sauter une chaîne*/
```

E/S avec format (2 / 3)

sprintf et sscanf

- Identiques à `printf` et `scanf` mais l'E/S se fait dans une chaîne au lieu d'un fichier

snprintf

- La taille de la chaîne résultat est passée en paramètre (plus sûr)
- Exemple

```
char str[MAXBUF];
int n;
snprintf(str, MAXBUF, "%3d error%s", n, n > 1 ? "s" : "");
```

E/S avec format (3 / 3)

vprintf et vscanf

- Entrée/Sortie avec une liste variable d'arguments

```
int vprintf(const char *format, va_list ap);
int vsnprintf(char *str, size_t size, const char *format, va_list ap);

int vscanf(const char *format, va_list ap);
int vsscanf(const char *str, const char *format, va_list ap);
```

Exemple

```
void message(const char *format, ...) {
  static int count = 0;
  va_list ap;

  va_start(ap, format);

  printf("Message %d: ", ++count);
  vprintf(format, ap);
}
```

Le type FILE (1 / 3)

- défini dans <stdio.h>
- Un objet de type FILE est un flux
  - supporte accès séquentiel et direct
  - accès par caractère
  - E/S bufferisées
- Ouverture d'un fichier:

```
FILE *fopen(const char *name, const char *mode);
/* mode  peut être "r", "w", "a",  ... */
```

- Fermeture d'un fichier

```
int fclose(FILE *fp);
/* renvoie 0 si OK et EOF sinon */
```

Le type FILE (2 / 3)

Lecture fichier

```
int fscanf(filep, format, pointer_expression_list);
int getc(filep);
int fgetc(filep);
char *gets(string);  /* allocation de la chaîne par l'appelant */
char *fgets(string, size, filep);
int fread(buffer, size, nelem, filep);
```

Écriture fichier

```
int fprintf(filep, format, pointer_expression_list);
int putc(character, filep);
int fputc(character, filep);
char *puts(string);
char *fputs(string, filep);
int fwrite(buffer, size, nelem, filep);
```

Le type FILE (3 / 3)

Accès direct

```
int fseek(FILE *stream, long offset, int whence);
long ftell(FILE *stream);
void rewind(FILE *stream);
```

Origine pour fseek: SEEK_SET, SEEK_CUR ou SEEK_END

Gestion des buffers

```
void setbuf(FILE *stream, char *buf);
int setvbuf(FILE *stream, char *buf, int mode, size_t size);
```

mode: _IOFBF, _IOLBF ou _IONBF

Exemple: le programme cat

```c
void copy(FILE *fp) {
  int c;    /* et pas char! */
  while ((c = getc(fp)) != EOF)
    putc(c, stdout);
}

int main(int argc, char *argv[]) {
  FILE *fp;
  if (argc == 1) copy(stdin);
  else {
    while (--argc) {
      if ((fp = fopen(*++argv, "r")) == NULL) {
        fprintf(stderr, "cannot open %s\n", *argv);
        exit(1);
      }
      else {
        copy(fp);
        fclose(fp);
      }
    }
  }
  return 0;
}
```