

Système

Entrée-sortie

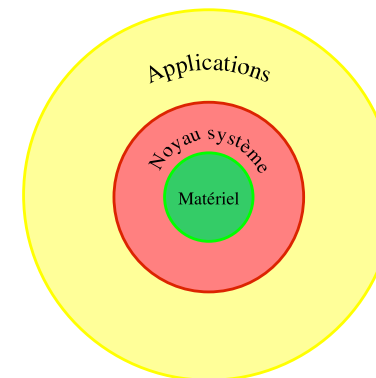
Gilles Roussel
Gilles.Roussel@univ-mlv.fr

Licence 2

6 février 2011

Les couches

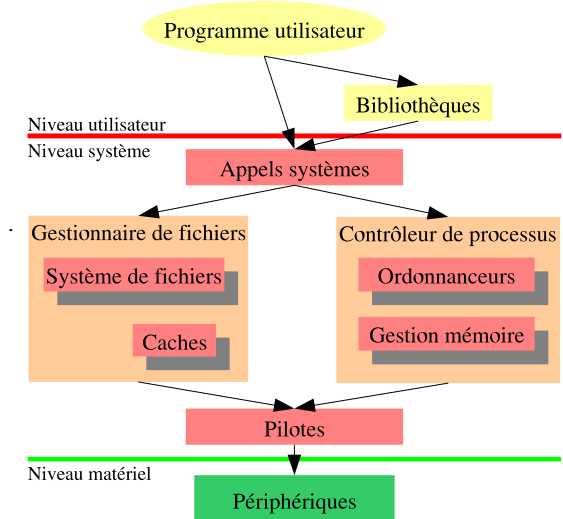
La ligne rouge est l'interface système
définie de façon très **rigoureuse**



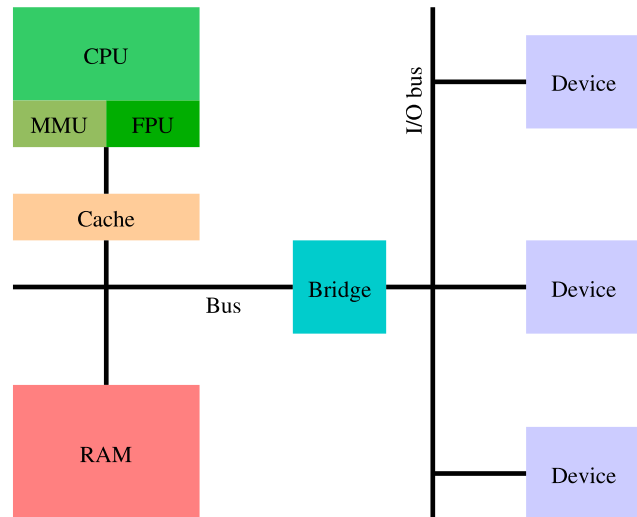
L'oignon

- Beaucoup de couches logicielles empilées
- Seuls les couches basses dépendent du matériel
- « Minimum » de chose à l'intérieur du noyau.
Critères :
 - sécurité
 - efficacité
- Programmation :
 - complexe
 - critique

Plus de détails !



C'est quoi le matériel ?



Gilles Roussel Système

Mais encore ?

- **Processeur :**
 - CPU (*Central Processing Unit*) gestion des instructions générales ;
 - MMU (*Memory Management Unit*) gestion de la mémoire et de la protection ;
 - FPU (*Floating Point Unit*) gestion des flottants ;Deux modes d'exécution : système et utilisateur. Alarme donne régulièrement le processeur au système
- **Mémoire :**
 - registres : petite mémoire à la vitesse du processeur ;
 - cache (anté-mémoire) : mémoire rapide de petite taille ;
 - RAM (*Random-Access Memory*) : mémoire centrale ;
- **Bus :** *medium* de communication de données ;
- **Bridge** (pont) : transferts entre différents type de bus ;
- **Device** (appareil ?) : disques, clavier, écran, souris, etc...

Gilles Roussel Système

Les entrées-sorties

- Bibliothèque standard (ANSI) de manipulation des entrées-sorties
- Type opaque de manipulation de descripteur **FILE *** défini dans **stdio.h**
- Descripteurs de base **stdin stdout** et **stderr**
- Descripteur associé (entre autre) à une position courante et à un tampon mémoire (*buffer*)
- Fonctions de manipulation *bufferisée* des entrées-sorties
 - **fgetc()** (getchar()) lecture d'un caractère
 - **fgets()** (~~getc()~~) lecture d'une chaîne de caractères
 - **fscanf()** (scanf()) lecture d'une chaîne de caractères formatée
 - **fputc()** (putchar()) écriture d'un caractère
 - **fputs()** (puts()) écriture d'une chaîne de caractères
 - **fprintf()** (printf()) écriture d'une chaîne de caractères formatée

Gilles Roussel Système

Parenthèse : « Quel fichier d'en-tête ajouter ? »

- Trouver le fichier .h grâce au manuel

```
# man printf
PRINTF(1)  User Commands  PRINTF(1)
NAME
    printf - format and print data
...
```
- Le bon !

```
# man -k printf
...
printf          (1) - format and print data
printf          (3) - formatted output conversion
....
# man 3 printf
PRINTF(3)  Linux Programmer's Manual  PRINTF(3)
NAME
    printf, ...
SYNOPSIS
    #include <stdio.h>
...
```

Gilles Roussel Système

Bufferisation en entrée

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

- Même quand un seul caractère est lu, plusieurs peuvent être lus et placés dans un tampon mémoire
- Entrée standard surtout bufferisée par le terminal de contrôle
 - Permet d'effacer les erreurs sans envoyer au programme le caractère erroné et le caractère d'effacement
 - Permet d'interpréter différemment certains caractères « spéciaux » (Ctrl-C, Ctrl-Z, etc.)
- **stty** permet de connaître et de changer le comportement du terminal de contrôle
 - **stty -a** affiche les caractéristiques du terminal de contrôle
 - **stty -icanon** permet de passer le terminal de contrôle en mode non-bufferisé
 - **stty icanon** permet de passer le terminal de contrôle en mode bufferisé

Gilles Roussel

Système

Bufferisation en sortie

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

- Données d'abord écrites dans un tampon mémoire
- Tampon mémoire vidé quand un caractère fin de ligne (`\n`) est écrit ou quand le tampon mémoire est plein
- Par défaut :
 - sortie standard sur terminal bufferisée par ligne
 - sortie d'erreur sur terminal non-bufferisée
- **fflush()** permet de vider le tampon mémoire
- **setvbuf()** permet de changer le comportement par défaut (avant toute utilisation)
 - **_IONBF** non bufferisé
 - **_IOLBF** bufferisé par lignes
 - **_IOFBF** bufferisation complète
- **setvbuf()** permet de préciser le tampon mémoire à utiliser

Gilles Roussel

Système

Bufferisation en sortie

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

stdio_test.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    fprintf(stdout, "stdout");
    fprintf(stderr, "stderr");
    fprintf(stdout, "\n");
    exit(EXIT_SUCCESS);
}
```

```
# ./stdio_test
stderrstdout
```

Gilles Roussel

Système

Bufferisation en sortie

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

stdio_test2.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    fprintf(stdout, "stdout");
    fflush(stdout);
    fprintf(stderr, "stderr");
    fprintf(stdout, "\n");
    exit(EXIT_SUCCESS);
}
```

```
# ./stdio_test2
stdoutstderr
```

Gilles Roussel

Système

Bufferisation en sortie

Système

Gilles Roussel

Le système

Bufferisation

Appels
systèmes

Parenthèses

Lignes

Tabs

Fichiers

stdio_test3.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    setvbuf(stdout, (char *)NULL, _IONBF, 0);
    fprintf(stdout, "stdout");
    fprintf(stderr, "stderr");
    fprintf(stdout, "\n");
    exit(EXIT_SUCCESS);
}
```

```
# ./stdio_test3
stdoutstderr
```

Gilles Roussel

Système

Bufferisation en sortie

Système

Gilles Roussel

Le système

Bufferisation

Appels
systèmes

Parenthèses

Lignes

Tabs

Fichiers

stdio_test4.c

```
#include <stdio.h>
#include <stdlib.h>
static char buffer[1024];

int main(int argc, char *argv[]) {
    setvbuf(stderr, buffer, _IOFBF, 1024);
    fprintf(stdout, "stdout");
    fprintf(stderr, "stderr");
    fprintf(stdout, "\n");
    exit(EXIT_SUCCESS);
}
```

```
# ./stdio_test4
stdout
stderr
```

Gilles Roussel

Système

Pourquoi bufferiser ?

Système

Gilles Roussel

Le système

Bufferisation

Appels
systèmes

Parenthèses

Lignes

Tabs

Fichiers

- Pas de bufferisation :
 - Affiche le plus vite possible les données soumises
- Par ligne :
 - Évite que les lignes affichées par plusieurs processus se mélangent
- D'une taille donnée
 - Améliore les performances

Gilles Roussel

Système

Améliorer les performances ?

Système

Gilles Roussel

Le système

Bufferisation

Appels
systèmes

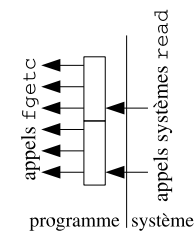
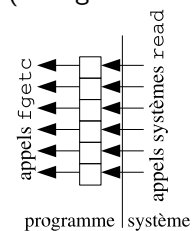
Parenthèses

Lignes

Tabs

Fichiers

- Pour lire (ou écrire) besoin d'accéder au matériel
- Programme n'y accède pas directement
- Besoin de « passer la main » au système
 - Réalisé via un **appel système** (read() ou write())
 - Entraîne un **changement de contexte** qui est coûteux
- Bufferisation évite de multiples appels systèmes (changements de contexte)



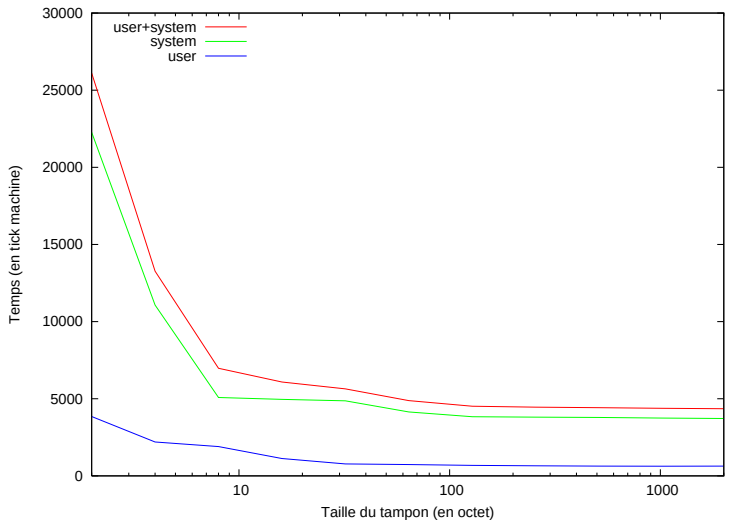
Gilles Roussel

Système

Un benchmark

Système	
Gilles Roussel	
Le système	
Bibliothèque	
Appels systèmes	
Parenthèses	
Lignes	
Tabs	
Fichiers	
	<pre>#include <stdlib.h> #include <stdio.h> #include <sys/times.h> #include <string.h> #define MAX_BUFFER_SIZE 3000 int main(int argc, char * argv[]) { struct tms start, end; int i; char buffer[MAX_BUFFER_SIZE]; int size; FILE *file; FILE *from=fopen("/dev/urandom","r"), *to=fopen("/dev/null","w"); if (!access("bench",F_OK)) { fprintf(stderr,"bench file already exists!\n"); exit(EXIT_FAILURE); } file = fopen("bench","w"); for (size=2;size < MAX_BUFFER_SIZE; size*=2) { setvbuf(from,buffer,_IOFBF,size); setvbuf(to,buffer,_IOFBF,size); times(&start); for (i=0;i<100000000;i++) { fputc(fgetc(from),to); } times(&end); fprintf(file,"%d %ld %ld %ld\n",size, end.tms_utime-start.tms_utime,end.tms_stime-start.tms_stime, end.tms_etime-start.tms_etime-start.tms_stime); fflush(file); } fclose(file);fclose(from); fclose(to); exit(EXIT_SUCCESS); }</pre>
Gilles Roussel	Système

Un benchmark

Système	
Gilles Roussel	
Le système	
Bibliothèque	
Appels systèmes	
Parenthèses	
Lignes	
Tabs	
Fichiers	
	
Gilles Roussel	Système

Parenthèse : « Le fichier résultat »

Système	
Gilles Roussel	
Le système	
Bibliothèque	
Appels systèmes	
Parenthèses	
Lignes	
Tabs	
Fichiers	
	<pre>2 3845 22269 26114 4 2196 11074 13270 8 1899 5078 6977 16 1122 4960 6082 32 773 4868 5641 64 733 4146 4879 128 679 3833 4512 256 653 3804 4457 512 636 3786 4422 1024 629 3749 4378 2048 633 3717 4350</pre>
Gilles Roussel	Système

Parenthèse : « Le fichier gnuplot »

Système	
Gilles Roussel	
Le système	
Bibliothèque	
Appels systèmes	
Parenthèses	
Lignes	
Tabs	
Fichiers	
	<pre>#!/usr/bin/gnuplot set output "bench.eps" set terminal postscript eps color enhanced solid set key left top set xlabel "Taille du tampon (en octet)" set ylabel "Temps (en tick machine)" set xr [2:2000] set yr [0:30000] set logscale x set style line 1 linetype 1 linewidth 2 set style line 2 linetype 2 linewidth 2 set style line 3 linetype 3 linewidth 2 plot "bench" using 1:4 title "user+system" with line 1, \ "bench" using 1:3 title "system" with line 2, \ "bench" using 1:2 title "user" with line 3</pre>
Gilles Roussel	Système

Parenthèse : « Et les tests d'erreur ? »

Système

Gilles Roussel

Le système

Bibliothèques

Appels

systèmes

Parenthèses

Lignes

Tubes

Fichiers

- Il faut tester les valeurs de retour
- `fprintf()` retourne le nombre de caractères écrits et une valeur négative en cas d'erreur
- `feof()` retourne normalement 0 et EOF en cas d'erreur
- `ferror()` retourne 0 s'il n'y a pas eu d'erreur
- `clearerr()` permet de supprimer l'indicateur d'erreur
- Numéro d'erreur placé dans variable globale `errno`
- `strerror()` et `perror()` permettent respectivement d'obtenir une chaîne de caractères associée à l'erreur et l'afficher sur la sortie d'erreur

Gilles Roussel

Système

Parenthèse : « (f)printf est une source de bugs de sécurité »

Système

Gilles Roussel

Le système

Bibliothèques

Appels

systèmes

Parenthèses

Lignes

Tubes

Fichiers

read_mem.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
    printf(argv[1]);
    exit(EXIT_SUCCESS);
}
```

```
# ./read_mem test
test
# ./read_mem '%s%s%s%s'
(null)ht^bf^c2^bf^c3^af^c2^bf^c2^bf^[%@
```

Gilles Roussel

Système

Lire ligne par ligne

Système

Gilles Roussel

Le système

Bibliothèques

Appels

systèmes

Parenthèses

Lignes

Tubes

Fichiers

- Il ne faut pas utiliser `gets()`
- `fgets()` pas si facile à utiliser pour être sûr de lire une ligne non tronquée (entière)
`char *fgets(char *s, int size, FILE *stream);`
- Bibliothèque non-standard (GNU) `libreadline` permet de simplifier la lecture d'une ligne

Gilles Roussel

Système

Comment installer une nouvelle bibliothèque ?

Système

Gilles Roussel

Le système

Bibliothèques

Appels

systèmes

Parenthèses

Lignes

Tubes

Fichiers

- Suite de commandes standard
- Récupérer l'archive des sources : `readline-6.0.tar.gz`
- Décompresser et extraire l'archive avec la commande `tar` (z décompression, x extraction, v verbeux, f depuis un fichier):
`# tar zxvf readline-6.0.tar.gz`
....
- Configurer par rapport à la plate-forme :
`# ./configure`
....
Éventuellement en précisant un répertoire d'installation
`# ./configure --prefix=${HOME}/lib`
....
- Compiler :
`# make`
....
- Installer :
`# make install`
....

Gilles Roussel

Système

Utiliser readline

Système

Gilles Roussel

Le système

Bufferisation

Appels
système

Parenthèses

Lignes

Tubes

Fichiers

- `readline()` alloue automatiquement la ligne qu'il vient de lire
- Il faut désallouer les lignes !
- `readline()` fournit un ensemble de raccourcis pour la gestion de ligne
- `readline()` permet de sauver les lignes et de les rappeler

Gilles Roussel

Système

Un exemple

Système

Gilles Roussel

Le système

Bufferisation

Appels
système

Parenthèses

Lignes

Tubes

Fichiers

readline.c

```
#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    char *line;
    while(NULL!=(line = readline("prompt# "))) {
        if (*line) {
            add_history(line);
        }
        if (fprintf(stdout,"%s\n",line) < 0) {
            break;
        }
        free(line);
    }
    exit(EXIT_SUCCESS);
}
```

```
# gcc -o readline readline.c -I$HOME/lib/include -L$HOME/lib/lib -lreadline -ltermcap
# LD_LIBRARY_PATH=$HOME/lib ./readline
```

Gilles Roussel

Système

Créer un pipe en C

Système

Gilles Roussel

Le système

Bufferisation

Appels
système

Parenthèses

Lignes

Tubes

Fichiers

- `popen()` permet de lancer une commande et de récupérer un descripteur pour écrire vers l'entrée standard de la commande ou lire depuis la sortie standard de la commande
- Descripteurs bufferisés
- Comme `system()` la commande est lancée *via* un Shell
- `pclose()` permet de fermer le descripteur ainsi ouvert
- `pclose()` est bloquée tant que le processus lancé n'a pas terminé
- `pclose()` retourne la valeur de retour du processus

Gilles Roussel

Système

Un exemple

Système

Gilles Roussel

Le système

Bufferisation

Appels
système

Parenthèses

Lignes

Tubes

Fichiers

```
#define _GNU_SOURCE
#include<stdio.h>
#include<stdlib.h>
```

```
int main(int argc, char* argv[]) {
    FILE *to_command;
    int result;
    if (NULL == (to_command = popen("cat","w"))) {
        perror("popen");
        exit(EXIT_FAILURE);
    }
    if (fprintf(to_command,"Test\n")<0) {
        perror("fprintf");
        exit(EXIT_FAILURE);
    }
    if ((result=pclose(to_command)) == -1) {
        perror("pclose");
        exit(EXIT_FAILURE);
    }
    exit(result);
}
```

Gilles Roussel

Système

Entrées-sorties vers les fichiers

- Système**
- Gilles Roussel
- Le système
- Bufferisation
- Appels système
- Parentthèses
- Lignes
- Tabes
- Fichiers
- **fopen()** permet d'ouvrir un fichier
`FILE *fopen(const char *path, const char *mode);`
 - "r" lecture, "r+" lecture/écriture à partir du début du fichier
 - "w" écriture, "w+" lecture/écriture avec tronquage ou création du fichier
 - "a" écriture, "a+" lecture/écriture toujours à partir de la fin du fichier
 - Retourne un descripteur ouvert sur le fichier ou **NULL** en cas d'erreur
 - Descripteur sur fichier bufferisé par défaut : il faut faire `fflush()` pour assurer l'écriture sur disque
 - **fclose()** permet de fermer un descripteur ouvert ; son tampon est vidé automatiquement (`fflush()`)
 - Retourne 0 si pas de problème et **EOF** en cas d'erreur

Gilles Roussel Système

Lecture/écriture binaire

- Système**
- Gilles Roussel
- Le système
- Bufferisation
- Appels système
- Parentthèses
- Lignes
- Tabes
- Fichiers
- Rarement utiles dans le cas de entrées-sorties standards
 - Écriture d'un tableau contenant **nmemb** éléments de taille **size**
`size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`
 - Lecture bloquante d'un tableau de **nmemb** éléments de taille **size**
`size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);`
 - Retourne le nombre d'éléments lus ou écrits
 - Si valeur de retour inférieure à nombre demandé, erreur ou fin de fichier atteinte (pour `fread()`)

Gilles Roussel Système

Exemple : cat.c

Système

Gilles Roussel

Le système

Bufferisation

Appels système

Parentthèses

Lignes

Tabes

Fichiers

```
#include <stdio.h>
#include <stdlib.h>
#include <libgen.h>
#include <errno.h>
#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
    FILE *from; char buffer[BUFFER_SIZE]; int i; int n; int exit_status = EXIT_SUCCESS;
    if (argc<2) {
        fprintf(stderr, "usage: %s file1 [file2 ...]\n", basename(argv[0])); exit(EXIT_FAILURE);
    }
    for (i=1; i<argc;i++) {
        if (NULL == (from=fopen(argv[i], "r"))) {
            perror(argv[i]); exit_status = EXIT_FAILURE; continue;
        }
        while ((n=fread(buffer, sizeof(char), BUFFER_SIZE, from))!=0) {
            if (fwrite(buffer, sizeof(char), n, stdout) != n) { break; }
        }
        if (ferror(from)) {
            perror("read"); clearerr(from); exit_status = EXIT_FAILURE;
        }
        if (ferror(stdout)) {
            perror("write"); clearerr(stdout); exit_status = EXIT_FAILURE;
        }
        fclose(from);
    }
    exit(exit_status);
}
```

Gilles Roussel Système

Déplacement dans un fichier

- Système**
- Gilles Roussel
- Le système
- Bufferisation
- Appels système
- Parentthèses
- Lignes
- Tabes
- Fichiers
- **fseek()** permet de se placer à une position particulière dans un fichier
`int fseek(FILE *stream, long offset, int whence);`
 - **SEEK_SET** se déplace de offset par rapport au début du fichier
 - **SEEK_CUR** se déplace de offset par rapport à la position courante
 - **SEEK_END** se déplace de offset par rapport à la fin du fichier (valeur négative pour revenir en arrière)
 - On peut aller au delà de la fin du fichier ; le fichier change de taille si écriture
 - `fseek()` retourne 0 s'il n'y a pas eu d'erreur et -1 dans le cas contraire
 - **ftell()** retourne la position courante ou -1 en cas d'erreur

Gilles Roussel Système

Exemple : tail.c

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

```
#include <stdio.h>
#include <stdlib.h>
#include <libgen.h>
#define BUFFER_SIZE 1024

int main(int argc, char *argv[]) {
    FILE *file; int size, nb; char *end; char buffer[BUFFER_SIZE];
    char usage[] = "usage: %s file nb_byte\n";
    if (argc != 3) {
        fprintf(stderr, usage, basename(argv[0])); exit(EXIT_FAILURE);
    }
    size = strtol(argv[2], &end, 10);
    if ((*end != '\0') || (argv[2] == end)) {
        fprintf(stderr, usage, basename(argv[0])); exit(EXIT_FAILURE);
    }
    if (NULL == (file = fopen(argv[1], "r"))) {
        perror(argv[1]); exit(EXIT_FAILURE);
    }
    if (fseek(file, -size, SEEK_END) != 0) {
        perror(argv[1]); exit(EXIT_FAILURE);
    }
    while(0 != (nb = fread(buffer, sizeof(char), BUFFER_SIZE, file))) {
        if (fwrite(buffer, sizeof(char), nb, stdout) != nb) {
            perror("write"); exit(EXIT_FAILURE);
        }
    }
    if (ferror(file)) { perror("read"); exit(EXIT_FAILURE); }
    fclose(file);
    exit(EXIT_SUCCESS);
}
```

Gilles Roussel

Système

Lister les répertoires

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

- Un répertoire est un fichier un peu particulier ; il n'est pas possible de l'ouvrir avec l'API standard
- **opendir()** permet d'obtenir descripteur ouvert sur un répertoire
- Des appels successifs à **readdir()** permettent de lire les entrées du répertoire
- **readdir()** retourne une structure **dirent** qui contient un champs **d_name** qui contient le nom d'une entrée du répertoire
- Attention, l'espace mémoire associé à **d_name** peut être réutilisé à chaque appel.
- **closedir()** permet de fermer un descripteur ouvert sur un répertoire

Gilles Roussel

Système

Exemple : ls.c

Système

Gilles Roussel

Le système

Bufferisation

Appels

système

Parenthèses

Lignes

Tabs

Fichiers

```
#include <sys/types.h>
#include <dirent.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    DIR *dir;
    struct dirent *dirent;
    if ((dir = opendir(".")) == NULL) {
        perror("opendir"); exit(EXIT_FAILURE);
    }
    while((dirent = readdir(dir)) != NULL) {
        printf("%s\n", dirent->d_name);
    }
    if (closedir(dir) == -1) {
        perror("opendir"); exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

Gilles Roussel

Système