

```

linux/kernel/panic.c
Copyright (C) 1991, 1992 Linus Torvalds
*/

/*
 * This function is used through-out the kernel (and/or user-space)
 * to indicate a major problem.
 */

#include <linux/config.h>
#include <linux/module.h>
#include <linux/sched.h>
#include <linux/delay.h>
#include <linux/reboot.h>
#include <linux/notifier.h>
#include <linux/init.h>
#include <linux/sync.h>
#include <linux/syscalls.h>
#include <linux/interrupt.h>
#include <linux/mm.h>

int panic_timeout;
int panic_on_oops;
int tainted;

EXPORT_SYMBOL(panic_timeout);

struct notifier_block *panic_notifier_list;

EXPORT_SYMBOL(panic_notifier_list);

/* If (HP) is set, then the panic function will
 * panic the system. */
static int __init panic_setup(char *str)
{
    panic_timeout = simple_strtoul(str, NULL, 10);
    return 0;
}

__setup("panic=", panic_setup);

/**
 * panic - halt the system
 * @fmt: The text string to print
 *
 * Display a message, then perform the system shutdown.
 * The panic_notifier_list are called after the message is
 * printed. This function never returns.
 */
void panic(const char *fmt, ...)
{
    static char buf[1024];
    va_list args;
    if (defined(CONFIG_MAGIC_SYSRQ) && magic_sysrq && !in_interrupt())
        unsigned long caller = (unsigned long) __builtin_return_address(0);

    if (panic_timeout > 0)
        printk(KERN_EMERG "Kernel panic - not syncing: %s\n", fmt);
    else
        printk(KERN_EMERG "Kernel panic - not syncing: %s\n", fmt);
    flush_smp_buffer();
    va_start(args, fmt);
    vsprintf(buf, size(buf), fmt, args);
    va_end(args);
    printk(KERN_EMERG "Kernel panic - not syncing: %s\n", buf);
    flush_smp_buffer();

    if (panic_timeout > 0)
        panic_timeout--;
}

```

Chapitre 6

Système d'Exploitation UNIX

Mécanismes d'interprétation du Shell (Redirections & Tubes)

BOUKRI KHALIL

Types de Shell

Shell	Nom	Description
Bourne Shell	sh	Shell disponible sur toute plateforme UNIX
C shell	csh	Shell développé par BSD
Korn shell	ksh	Bourne Shell étendu par l'AT&T
Bourne Again Shell	bash	Version améliorée de sh et csh. Fourni le plus souvent avec Linux.
Zero Shell	zsh	shell avec beaucoup de fonctionnalités : typage, substitution et complétion très poussées
Tenex	tcsh	csh étendu
rc	rc	Implémentation pour UNIX du shell de Plan 9
es	es	Extension de rc

Caractères spéciaux de Shell

Caractères	Description
tabulation, espace	Délimiteur de mot
retour chariot	Fin de la commande à exécuter
&	Lance une commande en tâche de fond
:::	Séparateur de commande
*?[][^]	Substitution de noms de fichiers
&& !	Opérateurs booléens
' " \	Caractères de quotation
<><<>> ' <>< & > & << - >	Opérateurs de redirection d'entrées sorties
\$	Valeur d'une variable
#	Début de commentaires
(){}	Groupement de commande

Métacaractères: Quotation

' " ` \ changent la façon dont le Shell interprète les caractères spéciaux.

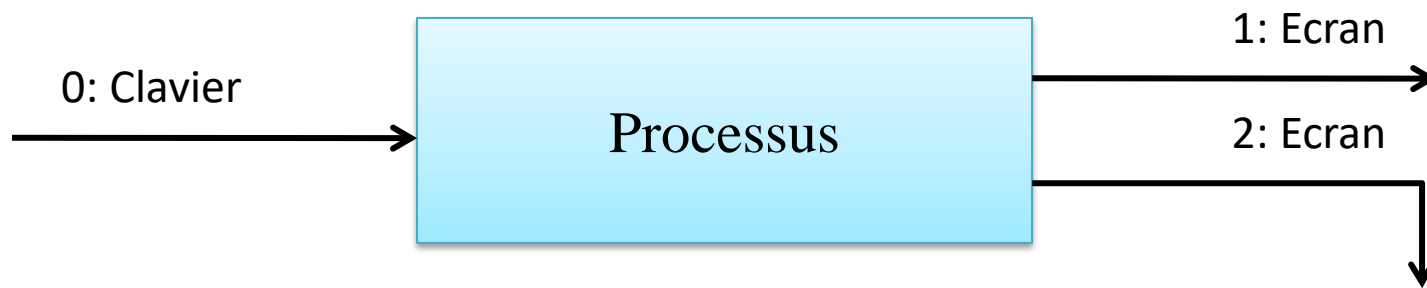
Symbole	Signification
' (single-quote)	le shell ignore tout caractère spéciaux entre deux '
" (double-quote)	le shell ignore tout caractère spéciaux entre deux ", à l'exception de \$ et \ et '
\ (antislash ou backslash)	le shell ignore le caractère spécial suivant le \
` (backquote ou antiquote)	le shell exécute ce qu'il y a entre deux `

Métacaractères: Quotation

Exemples

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[emsigl@localhost ~]$ whoami
emsigl
[emsigl@localhost ~]$ echo `whoami` et le chemin: $HOME et c*
emsigl et le chemin: /home/emsigl et cours
[emsigl@localhost ~]$
[emsigl@localhost ~]$
[emsigl@localhost ~]$ echo "`whoami` et le chemin: $HOME et c*"
emsigl et le chemin: /home/emsigl et c*
[emsigl@localhost ~]$
[emsigl@localhost ~]$ echo '`whoami` et le chemin: $HOME et c*'
`whoami` et le chemin: $HOME et c*
[emsigl@localhost ~]$
```

Flux: Redirections



- L'entrée standard (stdin : 0) : le flux d'entrée du programme (par défaut, ce qui est tapé au clavier)
- La sortie standard (stdout : 1) : le flux de sortie du programme (par défaut, il sera affiché à l'écran)
- L'erreur standard (stderr : 2) : le flux d'erreur du programme (par défaut, il sera affiché à l'écran)

Métacaractères de redirection

Syntaxe	Commande
<code>cmd < fic</code>	l'entrée de la commande provient du fichier
<code>cmd << etq</code>	l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette
<code>cmd > fic</code>	la sortie de la commande est placé dans le fichier
<code>cmd >> fic</code>	la sortie de la commande est mise à la suite du fichier
<code>cmd 2 > fic</code>	redirige la sortie d'erreur de la commande dans le fichier
<code>cmd 2 >> fic</code>	redirige la sortie d'erreur de la commande à la suite du fichier
<code>cmd_1 cmd_2</code>	passer la sortie de la commande 1 comme entrée de la commande 2

Redirection de la sortie standard

Syntaxe:

cmd > fichier

☛ redirige la sortie de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.

cmd >> fichier

☛ redirige la sortie de la commande à la suite du contenu du fichier (qui doit exister).

Exemple:

```
$ echo "Premier contenu ajouté" > file1
```

```
$ cat file1
```

Premier contenu ajouté

```
$ echo "Deuxième contenu ajouté " >> file1
```

```
$ cat file1
```

Premier contenu ajouté

Deuxième contenu ajouté

```
$ echo "Troisième contenu ajouté" > file1
```

```
$ cat file1
```

Troisième contenu ajouté

```
$
```


Redirection de la sortie d'erreur

Syntaxe:

cmd 2> fichier

☛ redirige la sortie d'erreur de la commande dans le fichier qui est créé s'il n'existait pas et dont le contenu est écrasé sinon.

cmd 2>> fichier

☛ redirige la sortie d'erreur de la commande à la suite du contenu du fichier (qui doit exister).

Exemple:

```
$ ls abc file1 file2
```

```
ls : abc : No such file or directory  
file1 file2
```

```
$ ls abc file1 file2 > sortiested
```

```
ls : abc : No such file or directory
```

```
$ cat sortiested
```

```
file1 file2
```

```
$ ls abc file1 file2 2> erreurstd
```

```
file1 file2
```

```
$ cat erreurstd
```

```
ls : abc : No such file or directory
```

```
$ ls abc file1 file2 2> erreurstd >sortiested
```

```
$ cat erreurstd
```

```
ls : abc : No such file or directory
```

```
$
```

Redirection de l'entrée standard

Syntaxe:

cmd < fichier

☛ l'entrée de la commande provient du fichier

cmd << ETQ

☛ l'entrée de la commande provient des lignes de commandes suivantes jusqu'à la ligne ne contenant que l'étiquette

Exemple:

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[emsig1@localhost ~]$ cat < liste_cours
php
linux
sql
[emsig1@localhost ~]$ 
[emsig1@localhost ~]$ 
[emsig1@localhost ~]$ cat <<fin
> bonjour
> je teste les redirections pour l'entree standard
> au revoir
> fin
bonjour
je teste les redirections pour l'entree standard
au revoir
[emsig1@localhost ~]$
```

Tube (Pipe) entre flux d'entrée et flux de sortie

Syntaxe:

cmd 1 | cmd 2

☛ redirige la sortie de la commande 1 vers l'entrée de la commande 2

**cmd1 | cmd2 <==> cmd1 > fictmp
 cmd2 < fictmp
 rm fictmp**

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[emsig1@localhost rep]$ cat file1
Bonjour
au revoir
[emsig1@localhost rep]$ cat file1 | wc
      2      3     19
[emsig1@localhost rep]$
```

Tube (Pipe) entre flux d'entrée et flux de sortie

Syntaxe:

cmd 1 | cmd 2

☛ redirige la sortie de la commande 1 vers l'entrée de la commande 2

La commande « xargs »

Certaines commandes ne savent pas lire leur entrée standard:

→ ls , rm, cp, ln, mv

Exemple :

ls | rm aucun sens

→ ls | xargs rm