Cours Systeme d'exploitation et Programmation Systeme

Mendy Fatnassi

10 décembre 2020

Table des matières

Chapitre 1 : Introduction

1.1 Généralité

1.1.1 Fonction de Manipulation des Fichier

Ouverture/Fermeture

open : Ouverture d'un fichier.
*int open(const char *nom, int oflag, mode_t mode)*
− oflag precise le mode d'acces au fichier :
O_RDONLY : *ouverture en lecture seule*
O_WRONLY : *ouverture en criture seule*
O_RDWR : *ouverture en lecture et criture*
O_NDELAY : *ouverture non bloquante*
O_APPEND : *position nement en fin de fichier avant chaque criture*
O_CREAT : *cration du fichier s'il n'existe pas*
O_TRUNC : *ouverture avec troncature si le fichier existe*
O_EXCL : *ouverture exclusive (retourne un code d'erreur si le fichier existe dja lors d'une cration)*

− mode correspond au droit d'acces (rwx).
Lorsque l'on fait appele a open il nous retourne un entier appel descripteur de fichier qui fait reference au

***close** : Cette fonction referme le fichier dont le descripteur est fd. En cas de réussite, elle retourne 0, sinon elle retourne -1.*
int close(int fd) `#include<fcntl.h>`

Lecture

***read** : Cette fonction lit n octets dans le fichier dont le descripteur est fd et les place dans un buffer. En cas de réussite, elle renvoie le nombre d'octets transferés, sinon elle retourne -1, 0 definie la fin de fichier pppour read.*
ssize$_t$ *read*(*int fd, void * buffer, size$_t$ n*); `#include<fcntl.h>`
Exemple d'utilisation :

```
#include<fcntl.h>
```

```
int main(){
    char c;
    int fd;

    fd =  open("/etc/passwd", O_RDONLY);
    if(fd == -1){
        fprintf(stderr,"impossible d'ouvrir le fichier\n");
        exit(1);
    }

    while( read(fd, &c, 1) > 0 )
        putchar(c);

    close(fd);

    return 0;
}
\end{verbatim}

\texbf{Ecriture}\\
\\
\underline{\textbf{write}}: Cette fonction écrit n octets dans le fichier don
\emph{ssize_t write(int fd, const void *buffer, size_t n);} \verb+#include<fc
Exemple d'utilisation :\\
\\
\begin{verbatim}
#include<fcntl.h>

int main()
{
    char buf[]="hello world !\n";
    int fd;

    fd =  open("foo", O_CREAT | O_RDWR, 0644);
    write(fd, buf, sizeof buf);

    return 0;
}
\end{verbatim}
\\
\\
```

```
Difference entre read et fgets :\\
fgets est une fonction, read est un appel système\\
fgets est C standard, read ne l'est pas\\
fgets est stdio tamponnées, read ne l'est pas\\
fgets fonctionne avec un FILE *, read fonctionne avec un descripteur de fichier\\
fgets lit jusqu'à newline, read lit combien vous le dites\\
\\
\textbf{lseek}: Permet la modification de la position courante.\\
\underline{Déclaration}:  off_t lseek(int fd, off_t offset, int whence);  \\
Le champ whence peut prendre comme valeurs:\\
SEEK_SET(debut), SEEK_CUR(courant), SEEK_END(fin)\\
\\
On a son equivalent avec des FILE fseek().\\
\\
fopen,fclose,fprintf marche comme open,close... mais avec des fichiers de type FILE


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Verrous  %%%%%%%%%%%%%%%%%%%%%%%%%%%
\nexpage

\chapter{Verrous}

Sous certaine distribution Linux/Unix on a besoins de faire un chmod g+x, g+s fichie

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\section{Bloquant/Non-Bloquant}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\subsection{Verrou Bloquant}

Rend impossible la pose de tout autre verrou sur le fichier en question. Si la pose
\\
On pourra pas ecrire en meme temps, dans un fichier, le processus qui voudra acceder

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\subsection{Verrou Non-Bloquant}

Rend possible la pose de tout autre verrou sur le fichier en question.Ne bloque pas
```

```
\\
On pourra alors ecrire sur le meme fichier a partir de 2 processus différent,

\section{Partagé/Exclusif}
Fichier .lck .\\
Les verrous peuvent etre externes c-à-d que le verouillage se fait pas struct
\\
Une structure \textbf{externe} peux etre bloquant ou non-bloquant.\\
\\
Pose d'un verrou externe : while(open("F_verrou",O_CREAT | O_EXCL,0)==-1) &&
Levee d'un verrouu externe : unlnk("F_verrou");
\\
\begin{tabular}{|c|c|}
\hline
Avantage & Inconvenient \\ \hline
Simple a faire & Consomme temps CPU (attente active) \\
Moins de charge pour le S.E & Pas de type de verrou (lecture/ecriture) \\
Portable &  Pas de porté de verrou \\ \hline
\end{tabular}
\\

\\
Une structure \textbf{interne} peux etre partagé ou exclusif (bloquant/non-bl
Pose d'un verrou interne: remplissage des champs de la structure flock et uti
Levee d'un verrou: \\
\begin{verbatim}
nom_verrou.l_type=F_UNLCK;
fcntl(fd,operation,&nom_verrou);
\end{verbatim}
\\
\underline{Inconvenient}:Plus de charge pour le SGF, pas portable.\\
\\
\textbf{Partagé}: Compatible pour la pose de verrou de meme type.\\
\textbf{Exclusif}: Incompatible avec tout types de verrous.\\
\\
\\
On utilisera la structure flock et la fonction fcntl:
\begin{verbatim}
struct flock {
  short l_type; /*Type de verrou*/
  short l_whence; /*Positionnement*/
```

```
  short l_start; /*Position du debut*/
  short l_len; /*Longueur*/
  short l_pid /*Pid du processus proprietaire*/
}
\end{verbatim}
\\
Le champ l_type peut prendre les valeurs suivant:\\
\begin{tabular}{|c|c|}
\hline
Type & Verrou \\ \hline
F_RDLCK & Verrou partagé (lecture) \\
F_WRLCK & Verrou exclusif (ecriture) \\
F_UNLCK & Verrou a lever \\ \hline
\end{tabular}
\\
Le champ l_whence peut prendre les valeurs suivant:\\
- SEEK_SET => 0: debut\\
- SEEK_CUR => 1: courant\\
- SEEK_END => 2: fin\\
\\
-La fonction fnctl permet de manipuler un descripteur de fichier, notamment la pose
\emph{fcnt(int fd,int operation,struct flock *verrou);} \#include <fcntl.h>\\
\\
Le champs operation peut prendre les valeur suivant:\\
\begin{tabular}{|c|c|}
\hline
Operation & Description \\ \hline
F_GETLK & Test d'existance d'un verrou \\
F_SETLK & Pose verrou non-bloquant \\
F_SETLKW & Pose verrou bloquant \\ \hline
\end{tabular}
\\
Dans le cas d'un deadlock (inter-blocage) la fonction fcntl renvoie une erreur :\\
if((fcntl(fd,F_SETLKW,0666)==-1) && (errno==EDEADLOCK))



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\section{Imperatif/Consultatif}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\subsection{Imperatif}

Quand il y lecture/ecriture concurrente, càd qu'on essaye d'acceder a un fich

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\subsection{consultatif}

L'utilisateur consulte si il y a deja un verrou existant.\\




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Signaux  %%%%%%%%%%%%
\nexpage

\chapter{Signaux}

Les interruption systéme (IT).\\


\section{Tableau des signaux}

\begin{tabular}{|c|c|c|c|}
\hline
N & Nom & Evenement & Comportement par defaut \\ \hline
2 & SIGINT & Interruption(int) & terminaison\\
3 & SIGQUIT & Interruption(quit) & "core dumped"\\
9 & SIGKILL & terminaison & terminaison\\
10 & SIGBUS  & bus error & "core dumped"\\
11 & SIGSEGV & violation memoire & "core dumped"\\
14 & SIGALRM  & alarme horloge & terminaison\\
16 & SIGUSR1 & signal 1 pour les util. & terminaison "user signal 1"\\
27 & SIGUSR2 & signal 1 pour les util. & terminaison "user signal 2"\\
23 & SIGSTOP & demande de suspension & suspension\\
25 & SIGCONT & demande de reprise de proc. suspendu & ignorance \\ \hline
```

```
\end{tabular}
\\
Avec comme comportement par defaut : \\
\\
\underline{terminaison}: Le processus est arreté.\\
\\
\underline{"core dumped"}: Le processus est arreté et ine image mémoire (fichier cor
\\
\underline{ignorance}: Le signal est sans effet.\\
\\
\underline{suspension}: Le processus est mit en sommeil.\\
\\
\\

\section{Fonctionnement}

Le mecanisme de la prise en compte des signaux se situe dans le bloc de controle d'u
Il comporte 3 vecteurs de NSIG bits correspondant a NSIG signaux: Signaux recus, Mas
Le bloc de controle permet de prendr en compte un signal delivré au processus et d'a
\\
Un processus prend en compte les signaux qu'il a recus au moment ou il passe en mode
\\
Un signal peut etre: \textbf{pendant},\textbf{délivré},\textbf{bloqué},\textbf{ignor
\underline{\textbf{Remarque}}: SIGKILL,SIGSTOP et SIGCONT ne peuvent pas etre bloqué
\\

\subsection{Handler}

Fonction de déroutement ou aussi appelé handler, ce sont les fonctions a réaliser a
\underline{Declaration des handler}: void hand (int signal); \\
\\
Constante de handler prédéfini:\\
\textbf{SIG_DFL}: Associe le traitement par défaut au signal (\emph{cf.tableau des s
\textbf{SIG_IGN}: Permet d'ignorer le signal.\\

\section{vrac}

Envoie de signaux:\\
int kill (pid_t pid,int sig); //pid: Identificateur du processus \\
#include <sys/types.h> et #include <signal.h>
Renvoie : 0 reussite , -1 erreur.\\
```

```
\\
pid_t getpid(void): Recupere l'identificateur du processus.\\
#include <unistd.h>
\\
Un ensemble de signaux est représenter par le type sigset_t.\\
Voici les primitive de manipulation des ensembles de signaux:\\
\begin{tabular}{|c|c|}
\hline
Fonction & Effet \\ \hline
sigmask(n) & Donne le masque du signal n\\
sigemptyset(S) & S=0  vide l'ensemble de signaux\\
sigaddset(S,n) & S= S $\cup$ n   ajoutent ou suppriment respectivement le sig
sigdelset(S,n) & S= S-n\\ \hline
sigismember(S,n) & Vrai ssi n $\in$ S\\  teste si le signal signum est membre
sigorset(S1,S2) & S1= S1 $\cup$ S2\\
sigandset(S1,S2) & S1= S1 $\cap$ S2\\
sigdiffset(S1,S2) & S1= S1 - S2\\ \hline
\end{tabular}
\\
Primitive generale:\\
Délcaration: int sigprocmask(int op, const sigset_t *p_ens, sigset_t *p_ens_a
#include <signal.h>\\
Renvoie : 0 reussite , -1 erreur.\\
\\
Donnée des champs:\\
- \textbf{op}: Operation a faire sur M = masque du processus appelant et peut
\textbf{SIG_BLOCK}: M= M+p_ens\\
\textbf{SIG_UNBLOCK}: M= M-p_ens\\
\textbf{SIG_SETMASK}: M= p_ens\\
\\
- \textbf{p_ens}: Pointeur sur une structure qui contient le masque. Si NULL
- \textbf{p_ens_ancien}: Pointeur sur une structure dans laquelle la primitiv
\\
\\
Autre primitives:\\
- sighold : bloque un signal donné.\\
- sigrelse: débloque un signal donné.\\
- sigpause: débloque un signal donné et met le processus en attente de récept
- sigsuspend: Installe un nouveau masque puis attend l'arrivé d'un signal n'a
- sigpending: Donne l'ensemble des signaux pendnats.\\
\\
```

```
\\
La l'origine la manipulation des handler se faisait par la primitive signal :\\
Déclaration: void (*signal (int sig, void (*hand)(int)) )(int); #include <signal.h>\
Cette primitive fait de la fonction hand(int) le nouveau handler du signal sig pour
\\
La norme POSIX propose la primitive sigaction regroupant l'ensemble des fonctionnali
Déclartion: int sigaction (int sig, const struct sigaction * p_action, struct sigact
\\
\begin{verbatim}
struct sigaction {
  int sa_flags; /* options pour prise en compte */
  void (*_handler)(); /* handler du signal */
  sigset_t sa_mask; /* signaux a masquer a la prise en compte */
}
\end{verbatim}
\\
Le champ sa_flags peut prendre comme valeur:\\
- \textbf{SA_RESETHAND}: Le handler par défaut est réinstaller a la prise en compte.
- \textbf{SA_RESTART}: Appels systéme repris aprés l'interruption.\\
- \textbf{SA_NODEFER}: Signal non bloqué automatiquement pendant l'exécution du hand


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Processus  %%%%%%%%%%%%%%%%%
\nexpage

\chapter{Processus}



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Tube  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\nexpage

\chapter{Tube}
```