PLIKI

Operacje powiązane z systemem:

- funkcje biblioteczne
  - stand. biblioteki C, libc;
  - wykonywane jak wywołania innych funkcji;
  - mogą wywoływać inne funkcje lub wywołania systemowe)
- wywołania systemowe
  - zaimplementowane w jądrze (jądro przejmuje wykonanie programu)
  - np. niskopoziomowe funkcje I/O
  - niektóre wywołania – tylko dla root'a
  - ok. 200; /usr/include/asm/unistd.h

polecenie **strace**

- śledzenie wykonania programu
- wypisuje wywołania i sygnały

strace –o plik *polecenie*

Wywołania systemowe

access – sprawdza uprawnienia procesu do pliku

access(sciezka,*uprawnienia*)

uprawnienia:

R_OK

W_OK

X_OK

F_OK

fstat – pobiera informacje o pliku

pokrewne funkcje: lstat,stat

wypełnia strukturę **stat**

fsync

- zapisuje na dysk dane zapisane do pliku (bez buforowania)
- O_SYNC w open

sendfile

- kopiowanie (deskryptor-deskryptor)
- bez bufora

readlink

writev

- wektorowy zapis/odczyt

Funkcje I/O niskiego poziomu

wywołania systemowe - dostęp do sterowników urządzeń

- open
- read
- write
- close
- ioctl – specyficzne sterowanie urządzeniem

program – deskryptory plików (0, 1, 2,..)

- bez buforowania

stdio.h – stdout - buforowanie

stderr - nie

write

#include <unistd.h>
size_t write(int despliku, const void *bufor, size_t nbajtów)

Zwraca: liczbę zapisanych bajtów
0 – koniec pliku
-1 – błąd

read

#include <unistd.h>
size_t read(int despliku, void *bufor, size_t nbajtów)

zwraca: liczbę przeczytanych bajtów
0 – koniec pliku
-1 – błąd

open

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
int open(const char *ścieżka, int oflagi);
int open(const char *ścieżka, int oflagi, mode_t tryb);
```

Zwraca – deskryptor pliku (niepowtarzalny; najniższy nieużywany)
-1 – błąd; ustawia errno

oflagi: O_RDONLY
O_WRONLY
O_RDWR
O_APPEND
O_TRUNC
O_CREAT – konieczny 3 parametr -tryb
O_EXECL
O_SYNC

tryb: S_I{R,W,X}{USR,GRP,OTH} S_IRUSR|S_IXGRP

żądane prawa dostępu są ustawiane zależnie od umask

polecenie: **umask 037**

w C: **umask(S_IRWXO|S_IWXGRP)**

open(……O_CREAT|O_EXECL,…..)

- niepodzielne sprawdzenie + otwarcie
- blokowanie plików ->synchronizacja

close

#include <unistd.h>

int close(int despliku);

Uwalnia deskryptor pliku

Zwraca: 0, -1

OPEN_MAX (limits.h) – określa limit plików

ioctl

#include <unistd.h>

int ioctl(int despliku, int komenda,…);

wykonuje *komenda* na pliku wskazanym przez *despliku*

lseek – ustawia wskaźnik odczytu/zapisu deskryptora pliku

```
#include <unistd.h>
#include <sys/types.h>
off_t lseek(int despliku, off_t przesunięcie, int skąd);
skąd:    SEEK_SET
         SEEK_CUR
         SEEK_END
```

dup dup2 – zwielokrotnia deskryptor pliku

dup – przydziela najniższy nieużywany deskryptor

```
#include <unistd.h>
int dup(int despliku);
int dup2(int des1pliku, int des2pliku);
```

unlink – zmniejsza l. dowiązań do pliku
(korzysta z tego rm)

link – tworzy link twardy

symlink – tworzy link symboliczny

```
#include <unistd.h>
int unlink(const char *ścieżka);
int link(const char *ścieżka1, const char *ścieżka2);
int symlink(const char *ścieżka1, const char *ścieżka2);
```

chmod

```
#include sys/stat.h>
int chmod(const char *ścieżka, mode_t tryb);
```

Dostęp do katalogów

mkdir

```
#include <sys/stat.h>
int mkdir(const char *ścieżka, mode_t tryb);
```

rmdir chdir getcwd

```
#include<unistd.h>
int rmdir(const char *ścieżka)
int chdir(const char *ścieżka)
char *getcwd(char *bufor, size_t rozmiar)
```

opendir readdir telldir seekdir closedir

```
#include<sys/types.h>
#include<dirent.h>
DIR *opendir(const char *nazwa);
struct dirent *readdir(DIR *katwskaźnik);
                    dirent:
                            ino_t d_ino
                            char  d_name[]
long int telldir(DIR *katwskaźnik); ->pozycja w strumieniu katalogu
                                      (lokalizacja)
void seekdir(DIR *katwskaźnik, long int lokalizacja);
int closedir(DIR *katwskaźnik);
```

standardowa biblioteka we/wy **stdio.h**

- dostarcza interfejs do niskopoziomowych wywołań systemowych we/wy

odpowiednik deskryptora pliku niskiego poziomu – strumień (*stream*) –FILE*

stdin, stdout, stderr – odpowiadają deskryptorom 0, 1 2

fopen

fclose

fread

fwrite

fflush

fseek

fgetc, getc, getchar

fputc, putc, putchar

fgets, gets

printf, fprintf, sprintf

scanf, fscanf, sscanf

Błędy strumieni

ferror feof clearerr

```
#include<stdio.h>
int ferror(FILE *strumień);
int feof(FILE *strumień);
void clearerr(FILE *strumień);
```

Program kopiujący pliki

- niskopoziomowy: open, read, write
  - znak po znaku
  - blokami
- za pomocą fcji z stdio.h : fopen, fgetc, fputc
- za pomocą fcji z stdio.h : fopen, fgets, fputs
- za pomocą sendfile/socket

time prog – porównanie czasów

stdio – wewnętrzny bufor w strukturze FILE; wywołania systemowe niskiego poziomu wykonywane tylko po zapełnieniu bufora

strumień pliku a deskrytptor niskiego poziomu

```
#include <stdio.h>
int fileno(FILE *strumień);
FILE *fdopen(int despliku, const char *tryb);
```

fileno – zwraca deskryptor dla strumienia pliku lub -1

fdopen – jak fopen; zwraca nowy strumień pliku lub NULL

zewnętrzna zmienna **errno**

```
#include <errno.h>
extern int errno;
```

EPREM	niedozwolona operacja
ENOENT	nie ma takiego pliku
EEXIST	plik istnieje
EINVAL	niedozwolony argument

strerror – zwraca ciąg opisujący błąd

```
#include <string.h>
char *strerror(int numerbłędu);
```

perror – opis błędu poprzedza komunikatem s

```
#include <stdio.h>
void perror(const char *s);
```

fcntl

– daje możliwości działania na niskopoziomowych deskryptorach plików

#include <unistd.h>

int fcntl(int deskr, int polecenie, long arg);

Polecenie:
- F_DUPFD - dup
- F_GETFD - 2 arg.; zwraca znaczniki pliku
- F_SETFD - ustawia znaczniki (FD_CLOEXEC)

F_GETLK, F_SETLK, F_SETLKW - blokowanie obszarów plików -> synchronizacja (niskopoziomowe operacje we/wy)

/proc/locks

mmap

- mapowanie pamięci

  tworzy wskaźnik do obszaru pamięci związanego z plikiem o deskryptorze deskr

#include<sys/mman.h>

void *mmap(void *adres, size_t długość, int prot, int flagi, int deskr, off_t off);

int msync(void *adres, size_t długość, int znaczniki);

int munmap(void *adres, size_t długość);

msync – zapis zmian pamięć – plik

munmap – zwolnienie segmentu pamięci

void ***mmap**(void ***adres**, size_t **długość**, int **prot**, int **flagi**, int **deskr**, off_t **off**);

adres – konkretny adres pamięci; 0 – wskaźnik przydzielony automatycznie

długość – rozmiar danych do zmapowania(długość segmentu pamięci)

prot – prawa dostępu do segmentu pamięci:

PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE

flagi – opcje mapowania

off – offset w pliku

Kompresja

compress lista_naw_plików

plik zastapiony zostaje przez *plik.Z*

-v – komentarze o każdym pliku

uncompress lista

zcat plik.Z

kompresja

gzip lista_nazw_plików

gzip plik – plik zamieniony na plik.gz

gunzip lista **gzip** –d lista

gunzip plik - plik.Z; plik.gz

stopnie kompresji (1-9) domyślnie gzip -6

opcja –r katalog dla gzip, gunzip, gzip –d

opcja –c – standardowe wy (nie nadpisuje plików)

zcat plik.gz zcat plik

Archiwizacja

tar

- tworzy archiwum plików i katalogów
- umożliwia uaktualnianie
- umożliwia rozszerzanie
- archiwum – na dowolnym urządzeniu, w pliku na dysku
- cel – kopia zapasowa; łączenie np. w celu przesłania

konwencja – pliki archiwum *.tar

tar

tar opcje **f** nazwa_arch.**tar** lista

pliki we. nie są kasowane

opcje:

c – twórz archiwum
t – wylistuj zawartość archiwum
x – odzyskaj zawartość archiwum
r - dołącz plik do archiwum
u – uaktualnij
M – archiwum na wielu nośnikach
z – kompresja archiwizowanych plików programem gzip
nie da się ich uaktualniać ani listować zawartości (u,r,t)
v – wyświetlenie nazw plików
f – wskazuje nazwę pliku archiwum (/etc/default/tar)
- - standardowe we/wy
C – zmiana bieżącego katalogu
p – zachowanie właściciela i praw dostępu

```
tar cf arch.tar programy
tar cvf arch.tar programy
tar xf arch.tar
tar rf arch.tar programy2
tar uvf arch.tar programy
tar tvf arch.tar
tar tf arch.tar
tar cfM /dev/fd0 programy
tar xf /dev/fd0
tar czf arch.tar programy
tar xzf arch.tar
tar –cf - -C /kat katstary|tar –xvpf -
```

Archiwum skompresowane

tar cvf arch1.tar k*

tar tvf arch1.tar

gzip arch1.tar

arch1.tar.gz

1. gunzip arch1.tar.gz; tar –xvf arch1.tar
2. tar –xzvf arch1.tar.gz
3. gunzip -c arch1.tar.gz|tar –xvf-

(- dane we. pochodzą z potoku)

-c – wy na stdout

Skompresowane archiwum złożone ze skompresowanych plików

tar cvzf arch1.tar k*

tar tvf arch1.tar -- źle; (opcje u, r też niedopuszczalne)

gzip arch1.tar

arch1.tar.gz

1. gunzip arch1.tar.gz; tar –xzvf arch1.tar
2. tar –xzvf arch1.tar.gz --- źle
3. gunzip -c arch1.tar.gz|tar –xzvf-

Sprawdzanie miejsca na dysku

du – podaje wielkość obszaru zajętego

df – podaje obszar dostępny w systemie plików

Główna literatura do laboratorium:

but M.Mitchell, J.Oldham,A.Samuel Linux Programowanie dla zaawansowanych

ciuchcia N.Mattew, R.Stones Linux Programowanie

PLIKI

zadania: 1,2 - obowiązkowe; 3,4,5 - dodatkowe

1. sprawdzenie buforowania stdout, stderr
   - funkcje z stdio.h
   - niskopoziomowe
2. porównanie czasów kopiowania dużego pliku (time)
   - wywołania niskopoziomowe po znaku
   - wywołania niskopoziomowe blokami (1024B, 4096B)
   - funkcje z stdio.h po znaku (fgetc, fputc)
   - funkcje z stdio.h blokami (fgets, fputs) (1024B, 4096B)

3. utworzenie dużego pliku wypełnionego zerami, który nie zajmuje miejsca (lseek)
4. szukanie pliku wg. nazwy w drzewie katalogowym
   + wyświetlenie informacji o znalezionym pliku: typ, rozmiar

   parametry : 1 - nazwa pliku; 2 - nazwa katalogu startowego do szukania
5. zapis nieciągłych obszarów pamięci - wektorowy (writev)