|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №4 по курсу Операционные системы ­  студента группы М8О-207Б-18 Хренова Геннадия Николаевича , № по списку 28  Адреса www, e-mail, jabber, skype [khrenov.gena@yandex.ru](mailto:khrenov.gena@yandex.ru)  Работа выполнена: “7“ декабря 2019 г.  Преподаватель: Миронов Е. С.  Входной контроль знаний с оценкой  Отчёт сдан “ “ 20 г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Файловый системы, технология File mapping.
2. **Цель работы**: Освоение принципов работы с файловыми системами, обеспечение обмена данными между процессами посредством технологии «File mapping»
3. **Задание** (*вариант №* 16): Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или отображаемые файлы. Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы. На вход программе подается команда интерпретатора команд и имя файла. Программа должна перенаправить стандартный ввод команды с этого файла и вывести результат команды в стандартный выходной поток. Использование операций write и printf запрещено.
4. **Оборудование** (*лабораторное*):

ЭВМ Pentium , процессор 2.7 ГГц , имя узла сети с ОП 4096 МБ

НМД 2 ГБ. Терминал адрес . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор intel core i3 , ОП 3072 , НМД 50 ГБ. Монитор

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Unix , наименование версия

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Ubuntu версия 18.04LTS

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Для выполнения работы необходимы следующие команды:

* pid\_t fork() - создание дочернего процесса. Дочерний процесс возвращает 0, при ошибке -1.
* int execlp(const char \**path*, const char \**arg*, ... , (char\*)NULL ) **-** замена образа памяти процесса. Первый параметр, по соглашению, должен указывать на имя, ассоциированное с файлом, который надо запустить. Список параметров *должен* заканчиваться указателем null и, так как это функция с переменным числом аргументов, этот указатель должен приводиться к *(char \*) NULL*. Значение возвращается только в случае ошибки (-1).
* void \* mmap(void \**start*, size\_t *length*, int *prot* , int *flags*, int *fd*, off\_t *offset*) - отображает файлы или устройства в памяти. mmap отражает *length* байтов, начиная со смещения *offset* файла (или другого объекта), определенного файловым описателем *fd*, в память, начиная с адреса *start*. Последний параметр (адрес) необязателен, и обычно бывает равен 0. Настоящее местоположение отраженных данных возвращается самой функцией mmap, и никогда не бывает равным 0. Аргумент *prot* описывает желаемый режим защиты памяти. Параметр *flags* задает тип отражаемого объекта, опции отражения и указывает, принадлежат ли отраженные данные только этому процессу или их могут читать другие.
* int munmap(void \**start*, size\_t *length*) - снимает отображение
* pid\_t waitpid(pid\_t *pid*, int \**status*, int *options*) - ожидает завершения процесса

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

Программа получает при вызове файл, в котором лежит команда и название файла для выполнения команды. Далее открывается файл и делается отображение его содержимого в память. После чего создается дочерний процесс, считывает содержимое отображенного файла и и выполняет команду.

Тесты:

1) Подать на вход более двух аргументов

2) команда cat

3) команда head

4) команда ls

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/wait.h>

#include<sys/mman.h>

#include <sys/stat.h>

#include <string.h>

int main(int argc, char\* argv[]){

if (argc != 2) {

fprintf(stderr, "invalid input\n");

exit(2);

}

struct stat statbuf;

pid\_t pid;

int status;

int err;

int fd = open(argv[1], O\_RDWR | O\_CREAT);

if (fd == -1){

fprintf(stderr, "can`t open file\n");

}

if ( fstat(fd, &statbuf) < 0 ) {

fprintf(stderr,"fstat error\n");

}

char\* fmp = (char\*)mmap(0, statbuf.st\_size, PROT\_READ, MAP\_SHARED, fd, 0);

if (fmp == MAP\_FAILED){

fprintf(stderr,"error mmap");

}

if(close(fd) == -1){

fprintf(stderr, "can`t close file\n");

}

pid = fork();

if (pid < 0) {

fprintf(stderr, "prosses not created\n");

exit(1);

}else if (pid == 0){

char a[20] = {0};

char b[20] = {0};

int n = 0;

while (fmp[n] != '\n' && fmp[n] != EOF) {

a[n]= fmp[n];

n++;

}

n++;

int k = 0;

while (fmp[n] != '\n' && fmp[n] != EOF) {

b[k]= fmp[n];

n++;

k++;

}

if (execlp(a, a, b, (char\*)NULL) == -1){

fprintf(stderr, "exec eror\n");

}

}else if(pid > 0){

if (waitpid(pid, &status, 0) == -1){

fprintf(stderr, "waitpid error\n");

}

munmap(fmp, statbuf.st\_size);

}

return 0;

}

**strace**

execve("./a.out", ["./a.out", "help.txt"], 0x7ffd550ef218 /\* 57 vars \*/) = 0

brk(NULL) = 0x557483436000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=92729, ...}) = 0

mmap(NULL, 92729, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f8cb6683000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\34\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f8cb6681000

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f8cb6082000

mprotect(0x7f8cb6269000, 2097152, PROT\_NONE) = 0

mmap(0x7f8cb6469000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f8cb6469000

mmap(0x7f8cb646f000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f8cb646f000

close(3) = 0

arch\_prctl(ARCH\_SET\_FS, 0x7f8cb66824c0) = 0

mprotect(0x7f8cb6469000, 16384, PROT\_READ) = 0

mprotect(0x557483086000, 4096, PROT\_READ) = 0

mprotect(0x7f8cb669a000, 4096, PROT\_READ) = 0

munmap(0x7f8cb6683000, 92729) = 0

openat(AT\_FDCWD, "help.txt", O\_RDWR|O\_CREAT, 0114440) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=15, ...}) = 0

mmap(NULL, 15, PROT\_READ, MAP\_SHARED, 3, 0) = 0x7f8cb6699000

close(3) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f8cb6682790) = 5684

wait4(5684, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 5684

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=5684, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

munmap(0x7f8cb6699000, 15) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**9.Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. Замечание автора по существу работы
2. Выводы: Отображение файла на память предполагает установление соответствия между самим файлом и определённым участком памяти. Работа с участком памяти более знакома и удобна, так как есть доступ к каждому байту, плюс это быстрее прямой работы с файлами, которая требует много системных вызовов(а это время). С точки зрения взаимодействия между процессами mmap представляет общий участок памяти, а любая общая память — это опасное место, за которым требуется внимательно следить.

