Лабораторна робота №13

Тема: Системні змінні та функції обробки помилок системних викликів в Unix-подібних ОС.

Мета роботи: отримати практичні навички роботи із змінними середовища та обробкою помилок, які передаються операційною системою програмі.

Теоретичні відомості

Операційна система Linux надає кожній запущеній програмі середовище оточенняя. Під середовищем розуміється сукупність пар змінна-значення. Імена змінних і їх значення є рядками. По тому, що існує угода, змінні середовища записуються прописними буквами.

Інтерпретатор команд, як і будь-яка інша програма, має в своєму розпорядженні своє середовище. Наприклад, деякі змінні оточення SHELL-інтерпретатора: HOME, PATH, SHELL, MAIL і т.д.

Функція main() є першою функцією, визначеною користувачем (тобто явно описаною в початковому тексті програми), якої буде передане управління після створення відповідного оточення програми, що запускається на виконання. Традиційно функція main() визначається таким чином:

```
main(int argc,char *argv[],char *envp[]);
```

Тут масив envp[] містить покажчики на змінні оточення, передавані програмі. Кожна змінна є рядком виду им'я_змінноййй=значення_змінної.

Стандарт ANSI C визначає тільки два перші аргументи функції main() - argc і argv. Стандарт POSIX.1 визначає також аргументт envp, хоча рекомендує передачу оточення програми проводити через глобальну змінну environ:

```
extern char **environ;
```

Рекомендується слідувати останньому формату передачі для поліпшення переносимості програм на інші платформи UNIX.

Обробка помилок

Оскільки базовим способом полученяї послуг ядра є системні виклики, розглянемо детальніше обробку помилок в цьому випадку. Змінна errno визначена таким чином:

#include<errno.h>
external int errno;

Слід звернути увагу на те, що значення errno не обнуляється наступним системним викликом, що нормально завершився. Т. о., значення errno має сенс тільки після системного виклику, який завершився з помилкою.

Стандарт ANSI C визначає дві функції, що допомагають повідомити причину помилкової ситуації: strerror(3C) і perror(3C). Функція strerror(3C) має вигляд:

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

Функція приймає як аргументу errnum номер помилки і повертає покажчик на рядок, що містить повідомлення про причину помилкової ситуації.
Функція perror(3C) оголошена таким чином:
#include<errno.h>
#include<strdio.h>
void perror(const char *s);
Функція виводить в стандартний потік повідомлень про помилки інформацію про помилкову ситуацію, грунтуючись на значенні змінної errno. Сторока s, передавана функції, передує цьому повідомленню і може служити додатковою інформацією, наприклад, містячи назву функції або програми, в якій відбулася помилка.
Наступний приклад ілюструє використання цих двох функцій:

```
#include<errno.h>
#include<string.h>
#include<stdio.h>
main(int argc, char *argv[])
{
        fprintf(stderr,"ENOMEM: %s \n",strerror(ENOMEM));
        errno=ENOEXEC;
        perror(argv[0]);
        return 0;
}
```

Існує таблиця, в якій приведені найбільш загальні помилки системних викликів, включаючи повідомлення, які звичайно виводять функції strerror(3C) і perror(3C), а також їх короткий опис. Наприклад:

Код помилки і повідомлення	***Повідомлення***
ENOMEM Not enough space	При спробі запуску програми (exec(2)) розмір запрошуваної пам'яті перевищив максимально можливий в системі.
ENOEXEC Exec format error	Спроба запуску на виконання файлу, який має права на виконання, але не є файлом допустимого виконуваного формату.

Запустивши програму, ми отримаємо наступний результат:
$ a.out
ENOMEM: Not enough space
a.out: Exec format error

Завдання 1

Написати програму, яка виводить на екран монітора значення всіх змінних середовища оточення.

Завдання 2

Написати програму, яка вибірково виводить значення змінних середовища оточення і встановлює нові значення за бажанням користувача.

Файлові операції за допомогою системних викликів

Для користувачів ОС та прикладних програмістів дисковий простір надається у вигляді сукопності файлів, організованих у *файлову систему.*

Файл – це набір даних у файловій системі, доступ до якого здійснюється за іменем. Термін «файлова система» може вживатися для двох понять: принципу оргпнізації даних у вигляді файлів і конкретного набору даних (зазвичай відповідної частини диска), організованих відповідно до такого принципу.

Файлові системи розглядають на логічному і фізичному рівнях:

логічний рівень визначає зовнішнє подання системи як сукопності файлів (які звичайно перебувають у каталогах), а також виконання операцій над файлами і каталогами (створення, вилучення, тощо);

фізичний рівень визначає принципи розміщення структур даних файлової системи на диску або іншому пристрої.

У середовищі програмування UNIX існують два основні інтерфейси для файлового введення/ виведення:

1. інтерфейс системних викликів, що пропонує системні функції низького рівня, що безпосередньо взаємодіють з ядром операційної системи;
2. стандартна бібліотека введення/виведення, пропонуюча функції буферізованого введення/виведення.

Другий інтерфейс є «надбудовою» над інтерфейсом системних викликів, пропонуючої зручніший спосіб роботи з файлами.

Загальні відомості про файлові операції

Назвемо основні файлові операції, які звичайно надає ОС для використання у прикладних програмах.

Відкриття файла. Після відкриття файла процес може із ним працювати (наприклад, робити читання і записування). Відкриття файла зазвичай передбачає завантаження в оперативну пам’ять спеціальної структури даних – дескриптора файла, який визначає його атрибути та місце розташування на диску. Наступні виклики використовуватимуть цю структуру для доступу до файла.

Закриття файла. Після завершення роботи із файлом його треба закрити. При цьому структуру даних, створену під час його відкриття, вилучають із пам’яті. Усі дотепер не збережені зміни записують на диск.

Створення файла. Ця операція спричиняє створення на диску нового файла нульової довжини. Після створення файл автоматично відкривається.

Вилучення файла. Ця операція спричиняє вилучення файла і вивільнення зайнятого ним дискового простору. Вона зазвичай недопустима для відкритих файлів.

Читання з файла. Ця операція звичайно зводиться до пересилання певної кількості байтів із файла, починаючи із поточної позиції, у заздалегідь виділений для цього буфер пам'яті режиму користувача.

Записування у файл. Здійснюють із поточної позиції у файл із заздалегідь виділеного буфера. Якщо на цій позиуії вже є дані, вони будуть перезаписані. Ця операція може змінити розмір файла.

Переміщення покажчика поточної позиції. Перед операціями читання і записування слід визначити, де у файлі перебувають потрібні дані або куда треба їх записати, задавши за допомогою цієї операції поточну позицію у файлі. Зазаначимо, що якщо перемістити покажчик файла за його кінець, а потім виконати операцію записування, довжина файла збільшиться.

Отримання і завдання атрибутів файлів. Ці операції дають змогу зчитувати поточні значення всіх або деяких атрибутів файла або задавати для них нові значення.

Завдання 1

Написати програму, яка реалізує копіювання файлів за допомогою засобів POSIX.

Завдання 2

Написати програму, яка відображує інформацію про атрибути файла, ім'я якого задається через параметр командного рядка.

Завдання 3

Написати програму, яка виконує обхід каталогу в POSIX.

Завдання 4

Написати програму, яка відображує права доступу поточного процесу до заданого файла.

Контрольні питання:

1. Чим відрізняються системнівиклики від функцій бібліотеки загального призначення?
2. Поясніть призначення і наведіть формат системного виклику open()?
3. Як здійснити читання і записування файлів за допомогою системних викликів?
4. Яке призначення системного виклику lseek()?
5. Як отримати інформацію про атрибути файла?
6. Як виконати читання вмісту каталогу за допомогою системних викликів?
7. Що таке змінні середовища оточення?
8. Назвіть функції для читання і зміни змінних середовища, приведіть їх формат.