Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №2

по дисциплине «Операционные системы и системное программирование»

на тему «Понятие процессов.»

|  |  |  |
| --- | --- | --- |
| Выполнил  студент гр. 250505  Рудько Т.А. |  | Проверил  Басак Д.В. |

Минск 2024

# УСЛОВИЕ ЛАБОРАТОРНОЙ РАБОТЫ

## Разработать две программы ‒ parent и child.

Перед запуском программы parent в окружении создается переменная среды CHILD\_PATH с именем каталога, где находится программа child.

Родительский процесс (программа parent) после запуска получает переменные среды, сортирует их в LC\_COLLATE=C и выводит в stdout. После этого входит в цикл обработки нажатий клавиатуры.

Символ «+», используя fork(2) и execve(2) порождает дочерний процесс и запускает в нем очередной экземпляр программы child. Информацию о каталоге, где размещается child, получает из окружения, используя функцию getenv(). Имя программы (argv[0]) устанавливается как

child\_XX, где XX ‒ порядковый номер от 00 до 99. Номер инкрементируется родителем.

Символ «\*» порождает дочерний процесс аналогично предыдущему случаю, однако информацию о расположении программы child получает, сканируя массив параметров среды, переданный в третьем параметре функции main().

Символ «&» порождает дочерний процесс аналогично предыдущему случаю, однако информацию о расположении программы child получает, сканируя массив параметров среды, ука-

занный во внешней переменной extern char \*\*environ, установленной хост-средой при запуске (см. IEEE Std 1003.1-2017).

При запуске дочернего процесса ему передается сокращенное окружение, включающее набор переменных, указанных в файле, который передается родительскому процессу как параметр командной строки. Минимальный набор переменных должен включать SHELL, HOME, HOSTNAME,

LOGNAME, LANG, TERM, USER, LC\_COLLATE, PATH. Дочерний процесс открывает этот файл, считывает имена переменных, получает из окружения их значение и выводит в stdout.

Дочерний процесс (программа child) выводит свое имя, pid, ppid, открывает файл с набором переменных, считывает их имена, получает из окружения, переданного ему при запуске, их значение способом, указанным при обработке нажатий, выводит в stdout и завершается.

Символ «q» завершает выполнение родительского процесса.

Программы компилируются с ключами

-W -Wall -Wno-unused-parameter -Wno-unused-variable -std=c11 -pedantic

Для компиляции, сборки и очистки используется make.

### **ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ**

В данной программе реализованы следующие алгоритмы и решения:

**`parent.c`**

1. Установка LC\_COLLATE=C: В начале программы устанавливается локаль "C" для LC\_COLLATE, что влияет на сравнение строк.
2. Проверка аргументов командной строки: Проверяется количество аргументов командной строки. Если оно не равно 2, программа выводит сообщение об ошибке и завершается.
3. Вывод переменных окружения: Вызывается функция printEnviron(), которая сортирует и выводит текущие переменные окружения.
4. Обработка опций и запуск дочерних процессов: Вызывается функция choose\_options(), которая входит в цикл обработки нажатий клавиш. В зависимости от введенной опции ('+', '\*', '&'), запускается соответствующий дочерний процесс.

**`parent\_utils.c`**

1. Функция choose\_options(): Циклически считывает ввод пользователя, выполняя соответствующие действия в зависимости от введенной опции. Для опции '+' используется путь к папке из переменной окружения, для '\*' - извлекается из envp, для '&' - из глобальной переменной environ.
2. Функция createChildProcess(): Создает дочерний процесс с использованием функций fork() и execve(). Передает определенный путь, имя файла с переменными окружения и опцию в дочерний процесс.

**`environ\_utils.c`**

1. Функция compareStrings(): Используется в функции qsort() для сравнения строк с учетом текущей локали.
2. Функция get\_env\_value(): Получает значение переменной окружения по ее имени из массива envp.
3. Функция printEnviron(): Выводит и сортирует текущие переменные окружения.
4. Функция print\_custom\_environ(): Выводит значения переменных окружения, считанных из файла.
5. Функции createCustomEnviroment(): Создает массив переменных окружения на основе содержимого файла. Обрабатывает особые случаи для переменных HOSTNAME и LC\_COLLATE.

**`child.c`**

1. Проверка аргументов командной строки: Проверяется количество аргументов командной строки. Если оно не равно 3, программа выводит сообщение об ошибке и завершается.
2. Вывод информации о процессе и переменных окружения: В зависимости от переданной опции ('+', '\*', '&'), вызывается соответствующая функция для вывода переменных окружения.
3. Функции print\_custom\_environ(): Выводит значения переменных окружения, как в родительском, игнорируя текущие или включая текущие переменные окружения.

**ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА**

Проект лабораторной работы организован с учетом модульного подхода, что позволяет легко поддерживать, расширять и изменять код. В нем выделяются следующие модули:

1. parent.c
   * main(): Основная функция, обрабатывает командные аргументы, устанавливает локаль, выводит переменные окружения, вызывает функцию для обработки опций и создания дочерних процессов.
2. parent\_utils.c
   * choose\_options(): Обрабатывает опции, вводимые пользователем, и запускает соответствующие дочерние процессы.
   * createChildProcess(): Создает дочерний процесс и выполняет указанную программу с определенными аргументами и окружением.
3. environ\_utils.c
   * compareStrings(): Сравнивает строки с учетом текущей локали (используется для сортировки).
   * get\_env\_value(): Получает значение переменной среды по ее имени из массива envp.
   * printEnviron(): Выводит и сортирует переменные окружения.
   * print\_custom\_environ(): Выводит значения переменных окружения, считанных из файла.
   * createCustomEnviroment(): Создает массив переменных окружения на основе содержимого файла.
4. child.c
   * main(): Основная функция, обрабатывает командные аргументы, выводит информацию о процессе и переменные окружения в зависимости от опции.
5. environ\_utils.h
   * Заголовочный файл с прототипами функций для работы с переменными окружения.
6. parent\_utils.h
   * Заголовочный файл с прототипами функций для родительского процесса.

Такая структура проекта способствует легкости поддержки, тестирования и расширения функциональности программы. Каждый модуль выполняет свою конкретную задачу, что делает код более читаемым и понятным для разработчиков.

# Сборка проекта и использование

# Сборка проекта

1. Клонирование репозитория:

git clone https://github.com/ваш-пользователь/ваш-репозиторий.git

* Переход в каталог проекта:

cd ваш-репозиторий

* Компиляция проекта:

gcc -o parent parent.c parent\_utils.c environ\_utils.c

gcc -o child child.c environ\_utils.c

Использование проекта:

1. Запуск родительского процесса:

CHILD\_PATH=<путь/к/папке> ./parent <путь/к/файлу> [+, \*, &]

* + <путь/к/папке> - путь к папке, используемый для запуска дочерних процессов.
  + <путь/к/файлу> - путь к файлу с переменными окружения.

1. Выбор опций в родительском процессе:
   * + - Запуск дочернего процесса с указанным расположением из переменной окружения "CHILD\_PATH".
   * \* - Запуск дочернего процесса с указанным расположением из переменной окружения "CHILD\_PATH" и текущими переменными окружения.
   * & - Запуск дочернего процесса с указанным расположением из глобальной переменной окружения "environ".
2. Ввод команд в родительском процессе:
   * q - Завершение программы.
3. Вывод информации о процессе и переменных окружения в дочернем процессе:
   * Дочерний процесс выводит информацию о себе и переменные окружения в зависимости от выбранной опции.

Пример использования:

1. Сборка:

git clone https://github.com/ваш-пользователь/ваш-репозиторий.git

cd ваш-репозиторий

gcc -o parent parent.c parent\_utils.c environ\_utils.c

gcc -o child child.c environ\_utils.c

* Запуск:

CHILD\_PATH=/путь/к/директории ./parent /путь/к/файлу с переменными/ +

* Выбор опции:

После запуска родительского процесса выберите опцию +.

* Ввод команд:

В родительском процессе введите команду q для завершения программы.

* Результат:

Дочерний процесс выведет информацию о себе и переменные окружения из файла.

# Метод тестирования и результаты

# Методы тестирования:

1. Запуск родительского процесса:
   * Проверка, что родительский процесс корректно запускается с правильными аргументами.
   * Ввод неверного количества аргументов для проверки обработки ошибок.
2. Обработка опций в родительском процессе:
   * Ввод различных опций (+, \*, &) и проверка соответствующего запуска дочерних процессов.
   * Ввод неизвестной опции для проверки обработки ошибок.
3. Работа дочернего процесса:
   * Проверка, что дочерний процесс выводит информацию о себе и переменных окружения в соответствии с выбранной опцией.
   * Ввод неверного количества аргументов для проверки обработки ошибок.
4. Чтение переменных из файла:
   * Проверка корректного считывания переменных окружения из файла.
   * Ввод неверного пути к файлу для проверки обработки ошибок.

# Примеры результатов тестирования:

1. Тест запуска родительского процесса:
   * Успешный запуск с корректными аргументами.
   * Сообщение об ошибке при неверном количестве аргументов.
2. Тест обработки опций:
   * Корректный запуск дочерних процессов с выбранными опциями.
   * Сообщение об ошибке при вводе неизвестной опции.
3. Тест работы дочернего процесса:
   * Вывод корректной информации о процессе и переменных окружения в зависимости от выбранной опции.
   * Сообщение об ошибке при неверном количестве аргументов.
4. Тест чтения переменных из файла:
   * Успешное считывание переменных окружения из файла.
   * Сообщение об ошибке при неверном пути к файлу.