Задачи к курсу “Операционные системы” ФИТ НГУ, 1-й семестр.Draft

Правила сдачи задач

При подготовке задачи к сдаче убедитесь, что:

1. Код оформлен в едином стиле:
    a. однообразное именование функций, типв, переменных, макросов и т.д. (при этом допускается, что для каждого из этих объектов стиль именования свой);
    b. однообразные отступы;
    c. однообразная расстановка скобок;
    d. и т.д..
2. Проверены возвращаемые значения функций и системных вызовов и предусмотрена адекватная реакция на ошибки. Можно не проверять на возвращаемое значение функцию printf().
3. Все ресурсы выделенные явно, должны быть также явно освобождены. Не должно быть точек выхода из программы, где не освобождаются явно выделенные ресурсы.
4. Компиляция должна проходить без ошибок и предупреждений на максимальном уровне предупреждений.
5. Программа делает то что требуется.

Обязательно подготовьтесь к ответам на вопросы по той теме, на которую рассчитана задача.

В процессе сдачи, преподаватель может также потребовать изменить, добавить какую-либо функциональность или провести программный эксперимент.

Критерии оценки

На оценку “удовлетворительно” надо сдать по одной задаче из каждого раздела.
На оценку “хорошо” - по две задачи из каждого раздела.
На оценку “отлично” - по три задачи из каждого раздела.

Задачи

Компиляция, сборка, запуск

1. Написать программу hello.c, которая выводит фразу “Hello world”:
    a. получить исполняемый файл;
    b. посмотреть unresolved symbols (puts, printf) с помощью nm;

    c. посмотреть зависимости (ldd);
    d. запустить.
2. Написать статическую библиотеку с функцией hello_from_static_lib() и использовать ее в hello.c:
    a. посмотреть исполняемый файл на предмет того будет ли функция hello_from_static_lib() unresolved. Почему?
    b. где находится код этой функции?
3. Написать динамическую библиотеку с функцией hello_from_dynamic_lib() и использовать ее с hello.c:
    a. посмотреть состояние функции hello_from_dynamic_lib в получившимся исполняемом файле. Объяснить увиденное.
4. Написать динамическую библиотеку с функцией hello_from_dyn_runtime_lib() и загрузить ее в hello.c с помощью dlopen(3). Объяснить что происходит.

Системные вызовы

1. Проведите следующие эксперименты:
    a. запустите программу hello world из предыдущей задачи под strace:
        i. обратите внимание какие системные вызовы были вызваны в процессе исполнения программы. Чем обусловлено такое количество системных вызовов. Какой системный вызов используется для вывода “hello world”? Изучите этот вызов и разберитесь что он принимает и возвращает.
        ii. используйте этот сискол в программе hello world вместо printf(). Убедитесь что этот вызов присутствует в выводе strace.
        iii. напишите свою обертку над этим сисколом. Для этого используйте функцию syscall() из libc. Также проверьте вывод strace.
    b. Запустите под strace команду ‘wget kernel.org’ (если нет wget, используйте curl). Получите статистику использования системных вызовов порожденным процессом.
2. Разберитесь как устроена функция syscall(). Напишите код, который напечатает hello world без использования функции syscall().
3. Разберитесь как работает системный вызов ptrace(2) и напишите программу, которая породит процесс и выведет все системные вызовы дочернего процесса. (Можно решить эту задачу после изучения темы “Процессы”).

Файлы и Файловые системы

1. Написать программу, которая копирует каталог “задом наперед”. Программа получает в качестве аргумента путь к каталогу. Далее:
    a. Программа создает каталог с именем заданного каталога, прочитанного наоборот. Если задан каталог “qwerty”, то должен быть создан каталог “ytrewq”.
    b. Программа копирует все регулярные файлы из исходного каталога в целевой (пропуская файлы другого типа), переворачивая их имена и

содержимое. То есть с именами файлов поступаем также как и с именем каталога, а содержимое копируется начиная с последнего байта и до нулевого.

2. Написать программу, которая создает, читает, изменяет права доступа и удаляет следующие объекты: файлы, каталоги, символьные и жесткие ссылки. Для определения того какая именно функция должна быть исполнена предлагается иметь необходимое количество жестких ссылок на исполняемый файл с именами соответствующими выполняемому действию и в программе выполнять функцию соответствующую имени жесткой ссылки. Программа должна уметь:
    a. создать каталог, указанный в аргументе;
    b. вывести содержимое каталога, указанного в аргументе;
    c. удалить каталог, указанный в аргументе;
    d. создать файл, указанный в аргументе;
    e. вывести содержимое файла, указанного в аргументе;
    f. удалить файл, указанный в аргументе;
    g. создать символьную ссылку на файл, указанный в аргументе;
    h. вывести содержимое символьной ссылки, указанный в аргументе;
    i. вывести содержимое файла, на который указывает символьная ссылка, указанная в аргументе;
    j. удалить символьную ссылку на файл, указанный в аргументе;
    k. создать жесткую ссылку на файл, указанный в аргументе;
    l. удалить жесткую ссылку на файл, указанный в аргументе;
    m. вывести права доступа к файлу, указанному в аргументе и количество жестких ссылок на него;
    n. изменить права доступа к файлу, указанному в аргументе.

3. Написать программу, которая выводит содержимое /proc/pid/pagemap

Адресное пространство процесса

1. Структура адресного пространства.
    a. Напишите программу, которая создает переменные и выводит их адреса:
        i. локальные в функции;
        ii. статические в функции;
        iii. константы в функции;
        iv. глобальные инициализированные;
        v. глобальные неинициализированные;
        vi. глобальные константы.
    b. Сопоставьте адреса переменных с областями адресного пространства из соответствующего /proc/<pid>/maps. Объясните увиденное.
    c. Используя утилиту nm (или readelf) определите в каких секциях находятся выделенные переменные переменные.
    d. Напишите функцию, которая создает и инициализирует локальную переменную и возвращает ее адрес. Прокомментируйте результат и дайте оценку происходящему.
    e. Напишите функцию, которая:
        i. выделяет на куче буфер (например, размером 100 байт);

ii. записывает в него какую-либо фразу (например, hello world);
iii. выводит содержимое буфера;
iv. освобождает выделенную память;
v. снова выводит содержимое буфера;
vi. выделяет еще один буфер;
vii. записывает в них какую-либо фразу (например, hello world);
viii. выводит содержимое буфера;
ix. перемещает указатель на середину буфера;
x. освобождает память по этому указателю.
xi. выводит содержимое буфера;

f. Прокомментируйте работу предыдущего пункта.
g. Заведите переменную окружения.
h. Добавьте в вашу программу код, который:
    i. распечатывает ее значение;
    ii. изменяет его значение;
    iii. повторно распечатывает ее значение.
i. Запустите вашу программу и убедитесь что переменная окружения имеет требуемое значение.
j. Выведите значение переменной окружения после того как ваша программа завершилась.
k. Объясните произошедшее.

2. Управление адресным пространством:
    a. Напишите программу, которая:
        i. выводит pid процесса;
        ii. ждет одну секунду;
        iii. делает exec(2) самой себя;
        iv. выводит сообщение “Hello world”
    b. Понаблюдайте за выводом программы и содержимым соответствующего файла /proc/<pid>/maps. Объясните происходящее.
    c. Напишите программу, которая:
        i. выводит pid процесса;
        ii. ждет 10 секунд (подберите паузу чтобы успеть начать мониторить адресное пространство процесса, например, watch cat /proc/<pid>/maps);
        iii. напишите функцию, которая будет выделять на стеке массив (например, 4096 байт) и рекурсивно вызывать себя;
        iv. понаблюдайте как изменяется адресное пространство процесса (стек);
        v. напишите цикл, в котором на каждой итерации будет выделяться память на куче (подберите размер буфера сами). Используйте секундную паузу между итерациями.
        vi. понаблюдайте как изменится адресное пространство процесса (heap);
        vii. освободите занятую память.
        viii. присоедините к процессу еще один регион адресов размером в 10 страниц (используйте mmap(2) с флагом ANONYMOUS).
        ix. понаблюдайте за адресным пространством.

x. измените права доступа к созданному региону и проверьте какая будет реакция, если их нарушить:
    1. запретите читать данные и попробуйте прочитать из региона.
    2. запретите писать и попробуйте записать.
xi. попробуйте перехватить сигнал SIGSEGV.
xii. отсоедините страницы с 4 по 6 в созданном вами регионе.
xiii. понаблюдайте за адресным пространством.

d. Чтобы было удобнее наблюдать за адресным пространством подберите удобные паузы между операциями изменяющими его.
e. Объясните что происходит с адресным пространством в данной задаче.

3. Самодельная куча
    a. Реализуйте свою кучу над анонимным регионом адресов:
        i. присоедините анонимный регион (mmap(2));
        ii. реализуйте функцию my_malloc(), которая:
            1. принимает размер памяти в байтах;
            2. резервирует буфер запрошенного размера и возвращает указатель на его начало;
            3. при недостатке памяти возвращает NULL.
        iii. реализуйте функцию my_free(), которая:
            1. принимает указатель на буфер, возвращенный ранее функцией my_malloc();
            2. помечает буфер свободным;
        iv. Рекомендация. Для отладки можно присоединить регион связанный с файлом. Это позволит наблюдать за состоянием вашей кучи при выделении-освобождении памяти.

Создание, завершение процесса

1. Жизненный цикл процесса.
    a. Напишите программу, которая:
        i. создает и инициализирует переменную (можно две: локальную и глобальную);
        ii. выводит ее (их) адрес(а) и содержимое;
        iii. выводит pid;
        iv. порождает новый процесс (используйте fork(2)).
        v. в дочернем процессе выводит pid и parent pid.
        vi. в дочернем процессе выводит адреса и содержимое переменных, созданных в пункте a;
        vii. в дочернем процессе изменяет содержимое переменных и выводит их значение;
        viii. в родительском процессе выводит содержимое переменных;
        ix. в родительском процессе делает sleep(30);
        x. в дочернем процессе завершается с кодом “5” (exit(2)).
        xi. в родительском процессе дожидается завершения дочернего, вычитывает код завершения и выводит причину завершения и код

завершения если он есть. В каком случае кода завершения не будет?
   b. Объясните результаты работы программы.
   c. Понаблюдайте за адресными пространствами в procfs.
   d. Понаблюдайте за состояниями процесса в procfs или с помощью утилиты ps.
2. Процесс в состоянии зомби:
   a. Модифицируйте предыдущую программу так чтобы дочерний процесс становился зомби.
   b. Объясните какую проблему решает данное состояние.
   c. Может ли родительский процесс оказаться в состоянии зомби? Если да, то что в этом случае произойдет с дочерним? Смоделируйте эту ситуацию.
3. Создание процесса при помощи системного вызова clone(2):
   a. Используйте системный вызов clone(2) для создания процесса:
      i. Память для стека выделите при помощи mmap(2), таким образом, чтобы новый регион был связан с файлом и синхронизировался с ним при изменении. Т.е. при записи в новый регион, данные должны синхронизироваться с файлом.
      ii. Напишите две функции:
         1. Первая - точка входа для нового процесса.
         2. Вторая функция должна выделять на стеке массив со строкой “hello world” и рекурсивно вызывать себя. Глубина рекурсии пусть будет равна 10.
      iii. Вызовите вторую функцию из функции нового процесса и после ее исполнения завершите новый процесс.
   b. Исследуйте полученный файл:
      i. найдите строки “hello world”. Сколько их и почему столько?
      ii. найдите переменную счетчик, которая используется для ограничения рекурсии.
      iii. найдите адреса возврата из функций.

Межпроцессное взаимодействие

1. Взаимодействие процессов через разделяемую память:
   a. Создайте регион памяти размером в одну страницу при помощи системного вызова mmap(2), таким образом, чтобы он был общим для двух процессов.
   b. Пусть один процесс непрерывно пишет целые числа (типа unsigned int) от 0 до максимального значения (0, 1, … max) в данный регион. Когда он доходит до конца региона, сразу начинает писать сначала.
   c. Другой процесс читает из этого региона и проверяет, что числа идут последовательно. При обнаружении сбоя последовательности, выводит соответствующее сообщение в консоль.
   d. Объясните причину таких сбоев.
   e. Как вы думаете, что можно было бы сделать чтобы этих сбоев не было?

2. Взаимодействие процессов через pipe:
    a. Аналогично предыдущей задаче, организуйте передачу последовательности целых чисел (unsigned int) через pipe.
    b. Объясните почему при передаче через pipe не наблюдаются проблемы, которые возникали при передаче через разделяемую память.
3. Взаимодействие через доменные сокеты:
    a. Сделайте процесс - эхо сервер на доменном сокете. Задача эхо-сервера - принимать соединения от клиентов, читать все что они пишут и отправлять прочитанное им обратно.
    b. Сделайте клиента для взаимодействия.
    c. Убедитесь, что ваш эхо-сервер может взаимодействовать с несколькими клиентами

Сеть

1. UDP - эхо сервер:
    a. Сделайте UDP-сервер, который принимает данные от клиентов и пересылает их обратно клиенту.
    b. Напишите UDP-клиента, для теста UDP-сервера.
    c. Проверьте, что UDP-сервер, работает с несколькими клиентами.
2. TCP - эхо сервер:
    a. Сделайте TCP-сервер, который принимает соединения от клиентов на заданном ip и port.
    b. создает новый процесс, в котором:
        i. читает данные от клиента;
        ii. пересылает их ему обратно.
    c. Напишите TCP-клиента для проверки TCP-сервера.
    d. Проверьте, что TCP-сервер работает с несколькими клиентами и обрабатывает сессии в разных процессах.
3. Реализуйте задачу из пункта 2 при помощи мультиплексирования ввода-вывода poll(2)/select(2).

Понятие пользователя. Управление правами.

1. Использование SUID-бита для доступа к файлам:
    a. Создайте файл с правами “чтение только владельцу”.
    b. Напишите программу, которая выводит:
        i. содержимое этого файла.
        ii. реальный и эффективный идентификаторы пользователя
    c. Запустите программу из-под своего пользователя и из-под чужого.
    d. Объясните результат.
    e. Установите SUID-бит.
    f. Запустите программу из-под своего пользователя и из-под чужого.
    g. Объясните результат.