Индивидуальное задание
Вызов Си функций из монитора

Формат DWARF

DWARF (debugging with attributed record formats) - это формат отладочного файла, используемый многими компиляторами и отладчиками для поддержки отладки на уровне исходного кода. Это формат отладочной информации внутри объектного файла. DWARF описание программы представляет собой древовидную структуру, в которой каждый узел может иметь дочерние или родственные узлы. Узлы могут представлять типы, переменные или функции.

DWARF использует ряд записей отладочной информации (DIE) для определения низкоуровневого представления исходной программы. Каждая запись отладочной информации состоит из идентификационного тега и серии атрибутов. Запись или группа записей вместе содержат описание соответствующей сущности в исходной программе. Метка задает класс, к которому принадлежит запись, а атрибуты определяют специфические характеристики записи.

Постановка задачи

Для запуска Си функции из монитора, необходимо предварительно узнать тип возвращаемого значения и аргументов функции. Для этого необходимо было реализовать несколько дополнительных функций для расширения поддержки формата DWARF.

Результаты работы

Для определения типов данных использовались данные DWARF файла, сгенерированного при сборке ядра JOS.

Определение возвращаемого значения производилось путем полной развертке дерева DIE, где искалась запись, имеющая имя необходимой функции и тег DW_TAG_subprogram. Функции, которые имеют тип возвращаемого значения не void, имеют в записи DIE атрибут, ссылающийся на тип возвращаемого значения. Печать самого типа производилась при помощи дополнительно реализованной функции print_type.

Информация о формальных параметрах функции также находилась с помощью похожего алгоритма. Каждая запись о функции, имеющей параметры имеет несколько дочерних узлов, имеющих тег DW_TAG_formal_parameter, из атрибутов которых можно извлечь необходимую информацию о их типах. Функция arguments_by_fname

выводит все типы аргументов функции по имени, за исключением некоторых неподдерживаемых типов, таких как перечесления.

Алгоритм работы функции print_type заключается в последовательном обращении к записям, содержащим информацию о типе параметра. Например, запись с тегом DW_TAG_const_type может ссылаться как сразу на базовый тип, к которому относится, так и к записи с тегом DW_TAG_pointer_type, обозначающей, что данный параметр является указателем. Последней записью всегда будет являться либо базовый тип, либо структура, на которых будет заканчиваться печать типа.

Обратиться к реализованным функциям можно путем вызова функции монитора mon_types с помощью команды types.

Тестирование

Проверка работы программы производилась на стандартных функциях из заголовочных файлов ядра.

Объявление оригинальной функции	**Вывод команды монитора**
void *memset(void *dst, int c, size_t len)	K> types memset Return type: pointer void (pointer void , int , typedef of long unsigned int)
int strcmp(const char *s1, const char *s2);	K> types strcmp Return type: int (pointer const char , pointer const char)
int info_by_address(const struct Dwarf_Addrs *addrs, uintptr_t p, Dwarf_Off *store);	K> types info_by_address Return type: int (pointer const struct Dwarf_Addrs, typedef of long unsigned int , pointer typedef of long long unsigned int)
void user_mem_assert(struct Env *env, const void *va, size_t len, int perm);	K> types user_mem_assert Return type: void (pointer struct Env, pointer const void , typedef of long unsigned int , int)