Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчёт по утилите

по курсу «Системное программирование»

по теме «Исследование утилиты dump»

Выполнил студент гр. 13541/2:
Ерниязов Т.Е.

Проверил преподаватель:
Душутина Е. В.

Санкт-Петербург
2019 г.

1 Цель работы

Цель работы выполнить техническое задание.

2 Техническое задание

Необходимо познакомиться с утилитой dump, а также исследовать её и и выполнить заданную модификацию. Исследование утилиты dump заключается в выполнении всех пунктов, представленных ниже:

- понять для чего необходима данная утилита
- рассмотреть все особенности утилиты
- запустить утилиту с всевозможными ключами
- выполнить анализ исходного кода
- проанализировать источник данных утилиты
- определить за что отвечает каждый участок кода, выполнить декомпозицию

3 Характеристики системы

```
1  lorismelik@lorismelik-Aspire-Z5700:~/$ cat /proc/version
2  Linux version 4.15.0-29-generic (buildd@lgw01-amd64-057) (gcc version 7.3.0 (Ubuntu 7.3.0-16ubuntu3))
       #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018
3
4  lorismelik@lorismelik-Aspire-Z5700:~/$ gcc --version
5  gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0
6  Copyright (C) 2017 Free Software Foundation, Inc.
7  This is free software; see the source for copying conditions.  There is NO
8  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
9
10 lorismelik@lorismelik-Aspire-Z5700:~/$ strace -V
11 strace -- version UNKNOWN
12 Copyright (c) 1991-2018 The strace developers <https://strace.io>.
13 This is free software; see the source for copying conditions.  There is NO
14 warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
15
16 Optional features enabled: stack-unwind stack-demangle m32-mpers mx32-mpers
17
18 lorismelik@lorismelik-Aspire-Z5700:~/$ ltrace -V
19 ltrace version 0.7.3.
20 Copyright (C) 1997-2009 Juan Cespedes <cespedes@debian.org>.
21 This is free software; see the GNU General Public Licence
22 version 2 or later for copying conditions.  There is NO warranty.
```

4 Что такое dump?

Утилита dump на самом деле разделена на 2 утилиты dump и restore. Встроенные во Unix системы, системные утилиты Dump и Restore, являются одним из самых надежных и безопасных средств резервного копирования, это неотъемлемый инструмент любого системного администратора, призванный, если и не восстановить данные в полном объеме, то хотя-бы вернуть то, что было сохранено и смягчить возможные последствия от потери информации.

Команда dump, работает с блочными устройствами и деревом inode, умеет создавать, как полную резервную копию данных, так и инкрементные дампы, до 10 уровней, целого диска или любого, отдельно взятого раздела. Dump работает, даже если файловая система, бэкап которой вам необходимо сделать, в данный момент является "живой то есть, смонтирована и используется (как правило, так и есть), перед копирование делается снимок (snapshot) файловой системы, что-бы убедится, что в процессе работы утилиты, не было сделано никаких изменений. Кроме того, команда dump умеет сжимать данные. Dump умеет разделять резервную копию на куски указанной длины или по мере заполнения принимающего устройства.

По-умолчанию, если не было явно назначено место хранения резервной копии, dump создает ее в устройстве для хранения на магнитной ленте.

Команда restore, выполняет восстановление данных, из сохраненных ранее, программой dump, резервных копий. Например можно восстановить из резервной копии данных, удаленный по неосторожности, файл. Как и dump, Restore можно использовать для восстановления файлов по сети.

Обычный сценарий для команды restore, восстановление из резервной копии на пустой раздел, отформатированный с помощью утилиты newfs. Сначала восстанавливается полная резервная копия, после чего можно восстанавливать инкрементные архивы, в порядке их создания.

5 Поддерживаемые опции

- -level Уровни сохранения. На уровне 0, полное резервное копирование, гарантирует, что вся файловая система будет скопирована.Уровни выше 0, инкрементальное копирование, говорят dump'у копировать все файлы новые или модифицированные с момента последнего сохранения любого уровня. По умолчанию уровень 0.

- -a "авто размер". Запись на ленту будет происходить пока не будет возвращено указание конца носителя. Это лучше всего подходит для большинства современных ленточных накопителей и используется по умолчанию.

- -B records Количество килобайт на выходной том, исключая те которые являются числом умноженым на размер блоков, и следующие команды меньше чем это умножение. Эта опция предотвращает подсчет размера ленты, который основан на длине и плотности.)

- -b blocksize (размер блока) Количество килобайт на выходной блок. Используемый по умолчанию размер блока 10.

- -c Изменения настроек по умолчанию, используемых для ленточного картриджа, с плотностью 8000 bpi, и длинной 1700 футов.

- -A archive-file Архивирование таблицы содержимого дампа в указанном файле архива, которую будет использовать restore , чтобы определить, находится ли файл в файле дампа, который восстанавливается.

- -d density (плотность) Устанавливает плотность ленты. По умолчанию 1600BPI)

- -D file Задает путь к файлу, в котором хранится информация о предыдущих полных и инкрементных дампах. Расположение по умолчанию /etc/dumpdates.

- -e inodes Исключает inode из дампа. Параметр inodes представляет собой список номеров inode, разделенных запятыми.

- -E file Прочитать список inode, которые будут исключены из дампа из файла текстового файла. Файл должен быть обычным файлом, содержащим номера inodes, разделенных символами новой строки.

- -f file Записывает резервную копию в файл; файл может быть специальным файлом устройства, таким как /dev/st0, /dev/rsd1c, обычный файл или - (стандартный вывод). Несколько имен файлов могут быть заданы как один аргумент, разделенный запятыми. Каждый файл будет использоваться для одного тома дампа в указанном порядке; если для дампа требуется больше томов, чем указано количество имен, последнее имя файла будет использоваться для всех оставшихся томов после запроса изменения носителя. Если имя файла имеет вид host: file или user @ host: file dump записывает в именованный файл на удаленном хосте (который уже должен существовать, dump не создает новый удаленный файл) с помощью rmt.

- -F script Запускает скрипт в конце каждой ленты (кроме последней). Имя устройства и номер текущего тома передаются в командной строке. Сценарий должен возвращать 0, если дамп должен продолжаться без запроса смены ленты, и 1, если дамп должен продолжаться, но попросить пользователя сменить ленту. Любой другой код завершения приведет к отмене дампа.

- -I nr errors По умолчанию, дамп игнорирует первые 32 ошибки чтения в файловой системе, прежде чем запрашивать вмешательство оператора. С помощью этого флага можно изменить значение количества пропущенных ошибок. Это полезно при запуске дампа в активной файловой системе, где ошибки чтения просто указывают на несоответствие между проходами маппинга и дампинга. Значение 0 означает, что все ошибки чтения будут игнорироваться.

- -jcompression level Сжимает каждый блок, используя библиотеку bzlib. (Необязательный) параметр указывает уровень сжатия, который будет использовать bzlib. Уровень сжатия по умолчанию равен 2. Если указан необязательный параметр, между буквой опции и параметром не должно быть пробелов.

- -k Использет аутентификацию Kerberos для связи с удаленными серверами.
- -L label Предоставленная пользователем метка текстовой строки помещается в заголовок дампа, где инструменты, такие как restore и file , могут получить к ней доступ.
- -n Всякий раз, когда дамп требует внимания оператора, оповестите всех операторов аналогично утилите wall.
- -q Отменяет дамп немедленно, когда требуется внимание оператора, без предупреждения в случае ошибок записи, смены носителя и т.д.
- -Q file Включает поддержку быстрого доступа к файлам. Позиции ленты для каждого inode хранятся в файле файла, который используется restore (если вызывается с параметром -Q и именем файла) для непосредственного позиционирования ленты при восстановлении файла, над которым сейчас работает.
- -s feet Указание количество ленты, необходимое для определенной плотности.
- -S Оценка размера. Определяет объем пространства, необходимого для выполнения дампа, фактически не делая этого, и отображает примерное количество байтов, которое он займет.
- -T date Использует указанную дату в качестве начального времени для дампа вместо времени, определенного при просмотре в /etc/dumpdates.
- -u Обновляет файл /etc/dumpdates после успешного дампа.
- -v Во время дампа выводится дополнительная информация, которая может быть полезна в сеансах отладки.
- -W Сообщает для каких файловых систем надо произвести резервное копирование. Эта информация получена из файлов /etc/dumpdates и /etc/fstab.
- -y Сжимает каждый блок для записи с помощью библиотеки lzo.

Демонстрирование работы с некоторыми флагами:

```
lorismelik@lorismelik-Aspire-Z5700:/home$ sudo dump -0aL backup.dump  /home/lorismelik/Изображения
  DUMP: Date of this level 0 dump: Tue Nov 27 01:04:23 2018
  DUMP: Dumping /dev/sda5 (/ (dir home/lorismelik/Изображения)) to /dev/tape
  DUMP: Label: /backup.dump
  DUMP: Writing 10 Kilobyte records
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 2243 blocks.
  DUMP: Volume 1 started with block 1 at: Tue Nov 27 01:04:23 2018
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing /dev/tape
  DUMP: Volume 1 completed at: Tue Nov 27 01:04:23 2018
  DUMP: Volume 1 2230 blocks (2.18MB)
  DUMP: 2230 blocks (2.18MB) on 1 volume(s)
  DUMP: finished in less than a second
  DUMP: Date of this level 0 dump: Tue Nov 27 01:04:23 2018
  DUMP: Date this dump completed:  Tue Nov 27 01:04:23 2018
  DUMP: Average transfer rate: 0 kB/s
  DUMP: DUMP IS DONE
```

Рис. 1: Стандартная работы утилиты с флагами -0aL

```
lorismelik@lorismelik-Aspire-Z5700:~$ sudo dump -W
Last dump(s) done (Dump '>' file systems):
  /dev/sda5      (     /) Last dump: never
```

Рис. 2: Флаг -W

```
lorismelik@lorismelik-Aspire-Z5700:/home$ sudo dump -0avL backup.dump /home/lorismelik/Изображения
  DUMP: Date of this level 0 dump: Thu Jan 31 04:21:37 2019
  DUMP: Dumping /dev/sda5 (/ (dir home/lorismelik/Изображения)) to /dev/tape
  DUMP: Excluding inode 8 (journal inode) from dump
  DUMP: Excluding inode 7 (resize inode) from dump
  DUMP: Label: backup.dump
  DUMP: Writing 10 Kilobyte records
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 2006 blocks.
  DUMP: Volume 1 started with block 1 at: Thu Jan 31 04:21:37 2019
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping directory inode 2
  DUMP: dumping directory inode 262145
  DUMP: dumping directory inode 262146
  DUMP: dumping directory inode 393448
  DUMP: dumping directory inode 525665
  DUMP: dumping directory inode 525667
  DUMP: dumping (Pass IV) [regular files]
  DUMP: dumping regular inode 401204
  DUMP: dumping regular inode 401327
  DUMP: dumping regular inode 403202
  DUMP: dumping regular inode 404073
  DUMP: dumping regular inode 404182
  DUMP: dumping regular inode 404188
  DUMP: dumping regular inode 524805
  DUMP: Closing /dev/tape
  DUMP: Volume 1 completed at: Thu Jan 31 04:21:37 2019
  DUMP: Volume 1 1990 blocks (1.94MB)
  DUMP: 1990 blocks (1.94MB) on 1 volume(s)
  DUMP: finished in less than a second
  DUMP: Date of this level 0 dump: Thu Jan 31 04:21:37 2019
  DUMP: Date this dump completed:  Thu Jan 31 04:21:37 2019
  DUMP: Average transfer rate: 0 kB/s
  DUMP: DUMP IS DONE
```

Рис. 3: Включение режима отладки

```
lorismelik@lorismelik-Aspire-Z5700:/home$ sudo dump -0S  /home/lorismelik/Изображения
2054144
```

Рис. 4: Оценка размера

6 Используемые файлы

- /dev/st0 Устройство по умолчанию для резервного копирования
- /etc/dumpdates Записи о датах дампа
- /etc/fstab Таблица дампов: файловая система и частота
- /etc/mtab Таблица дампов: монтированные файловые системы

7 Структура исходного кода (по файлам)

- itime.c - файл исходного кода, отвечающий за работу с датами дампа
- main.c - файл исходного кода, содержащий обработку введенных данных и основной алгоритм работы программы.
- optr.c - файл исходного кода, отвечающий за взаимодействие утилиты с оператором во время дампа.
- traverse.c - файл исходного кода, отвечающий за опредение объектов которые будут копированы и их сжатие.
- unctime.c - файл исходного кода, содержащий дополнительные функции для работы с датой.
- tape.c - файл исходного кода, содержащий функции для записи файла на магнитный носитель.
- makefile - мейк файл, для сборки утилиты

8 Принцип работы утилиты

Утилита состоит из main файла, в котором происходит обработка введенных пользователм флагом и и далее описан полный алгоритм работы утилиты. Все специальные функции (получение времени дампа, запись на носитель и т.д.) описаны в отдельных файлах и они вызываются по мере необходимости:

- объявление необходимых структур данных для хранения информации
- объявление define-секций для более удобного использования кода
- вспомогательные методы
- метод int main(int argc, char **argv)
  - обработка всех флагов утилиты
  - установка параметров для дампа
  - определение объектов, для которых будет применено резервное копирование
  - их сжатие и записывание на носитель
  - запись о дампе в соответсвующие файлы

Основной алгоритм утилиты работает в 4 этапа: на первых двух он определяет какие файлы и директории будут скопированы, а на 3 и 4 производит собственно копирование. Основной структурой дампа является u spcl

```
union u_spcl {
    char dummy[TP_BSIZE];
    struct  s_spcl {
        int32_t c_type;         /* record type (see below) */
        int32_t c_date;         /* date of this dump */
        int32_t c_ddate;        /* date of previous dump */
        int32_t c_volume;       /* dump volume number */
        u_int32_t c_tapea;      /* logical block of this record */
        dump_ino_t c_inumber;       /* number of inode */
        int32_t c_magic;        /* magic number (see above) */
        int32_t c_checksum;     /* record checksum */
#ifdef  __linux__
        struct  new_bsd_inode c_dinode;
#else
#ifdef sunos
        struct  new_bsd_inode c_dinode;
#else
        struct  dinode  c_dinode;   /* ownership and mode of inode */
#endif
#endif
        int32_t c_count;        /* number of valid c_addr entries */
        union u_data c_data;        /* see above */
        char    c_label[LBLSIZE];   /* dump label */
        int32_t c_level;        /* level of this dump */
        char    c_filesys[NAMELEN]; /* name of dumpped file system */
        char    c_dev[NAMELEN];     /* name of dumpped device */
        char    c_host[NAMELEN];    /* name of dumpped host */
        int32_t c_flags;        /* additional information */
        int32_t c_firstrec;     /* first record on volume */
        int32_t c_ntrec;        /* blocksize on volume */
                int32_t c_extattributes;    /* additional inode info */
                int32_t c_spare[30];        /* reserved for future uses */
    } s_spcl;
} u_spcl;
```

Рис. 5: u spcl

Ниже представлен метод, который производит дамп блока, а также структура block context.

```
struct block_context {
  ext2_ino_t ino;
  blk_t *buf;
  int cnt;
  int max;
  int next_block;
};
```

```
static int
dumponeblock(UNUSED(ext2_filsys fs), blk_t *blocknr, e2_blkcnt_t blockcnt,
       UNUSED(blk_t ref_block), UNUSED(int ref_offset), void * private)
{
  struct block_context *p;
  e2_blkcnt_t i;

  p = (struct block_context *)private;
  for (i = p->next_block; i < blockcnt; i++) {
    p->buf[p->cnt++] = 0;
    if (p->cnt == p->max) {
      blksout (p->buf, p->cnt, p->ino);
      p->cnt = 0;
    }
  }
  p->buf[p->cnt++] = *blocknr;
  if (p->cnt == p->max) {
    blksout (p->buf, p->cnt, p->ino);
    p->cnt = 0;
  }
  p->next_block = blockcnt + 1;
  return 0;
}
```

Далее предоставлена функция 1 этапа дампа. Здесь происходит проход по списку индексных дескрипторов файловой системы, чтобы найти все индексные дескрипторы, которые были изменены с момента предыдущего дампа. Также находит все директории в файловой системе для второго этапа.

```
int
mapfilesfromdir(UNUSED(dump_ino_t maxino), long long *tapesize, char *directory)
{
  errcode_t retval;
  struct mapfile_context mfc;
  ext2_ino_t dir_ino;
  char dir_name [MAXPATHLEN];
  int i, anydirskipped = 0;

  /*
   * Mark every directory in the path as being dumped
   */
  for (i = 0; i < (int)strlen (directory); i++) {
    if (directory[i] == '/') {
      strncpy (dir_name, directory, i);
      dir_name[i] = '\0';
      retval = ext2fs_namei(fs, ROOTINO, ROOTINO, dir_name,
                &dir_ino);
      if (retval) {
        com_err(disk, retval, "while translating %s",
          dir_name);
        exit(X_ABORT);
      }
      mapfileino(dir_ino, 0, tapesize, &anydirskipped);
    }
  }
  /*
   * Mark the final directory
   */
  retval = ext2fs_namei(fs, ROOTINO, ROOTINO, directory, &dir_ino);
  if (retval) {
    com_err(disk, retval, "while translating %s", directory);
    exit(X_ABORT);
  }
  mapfileino(dir_ino, 0, tapesize, &anydirskipped);

  mfc.tapesize = tapesize;
  mfc.anydirskipped = &anydirskipped;
  retval = ext2fs_dir_iterate(fs, dir_ino, 0, NULL, mapfilesindir,
            (void *)&mfc);

  if (retval) {
    com_err(disk, retval, "while mapping files in %s", directory);
    exit(X_ABORT);
  }
  /*
   * Ensure that the root inode actually appears in the file list
   * for a subdir
   */
  mapfileino(ROOTINO, 0, tapesize, &anydirskipped);
  /*
```

```
52      * Restore gets very upset if the root is not dumped,
53      * so ensure that it always is dumped.
54      */
55     SETINO(ROOTINO, dumpinomap);
56     return anydirskipped;
57   }
```

Здесь показана функция для получения даты дампа из файла dumpdates для определенной файловой системы.

```
1   void
2   getdumptime(int createdumpdates)
3   {
4     struct dumpdates *ddp;
5     int i;
6
7   #ifdef FDEBUG
8     msg("Looking for name %s in dumpdates = %s for level = %s\n",
9       disk, dumpdates, level);
10  #endif
11    spcl.c_ddate = 0;
12    memset(&lastlevel, 0, NUM_STR_SIZE);
13
14    /* If this is a level 0 dump, and we're not updating
15       dumpdates, there's no point in trying to read
16       dumpdates.  It may not exist yet, or may not be mounted.  For
17       incrementals, we *must* read dumpdates (fail if it's not there!) */
18    if ( (!strcmp(level, lastlevel)) && !createdumpdates)
19      return;
20    initdumptimes(createdumpdates);
21    if (ddatev == NULL)
22      return;
23    /*
24     *  Go find the entry with the same name for a lower increment
25     *  and older date
26     */
27    ITITERATE(i, ddp) {
28      if (strncmp(disk, ddp->dd_name, sizeof (ddp->dd_name)) != 0)
29        continue;
30      if (ddp->dd_level >= atoi(level))
31        continue;
32      if (ddp->dd_ddate <= (time_t)spcl.c_ddate)
33        continue;
34      spcl.c_ddate = ddp->dd_ddate;
35      snprintf(lastlevel, NUM_STR_SIZE, "%d", ddp->dd_level);
36    }
37  }
```

9 Вывод

В ходе выполнения данной работы познакомился с утилитой dump. Также была проверена работа всевозможных ключей и флагов утилиты. Кроме того был исследован исходной код утилиты. Прежде всего были внесены некоторые комментарии в исходный код для более понятного чтения. В итоге хотелось бы отметить то, что *dump* – удобная утилита для создания резервной копии файловой системы, поддерживающая диалог с оператором.