ISSN: 2321-7782 (Online)
Volume 1, Issue 7, December 2013
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Partition Table Analyzer for Cloud OS

Perla RaviTheja[1]
Research Scholar
School of Computing Science & Engineering
VIT University
Vellore - India

Varun Manchikalapudi[2]
Research Scholar
School of Computing Science & Engineering
VIT University
Vellore - India

SK. Khadar Babu[3]
Assistant Professor (Sr)
School of Advanced Sciences
VIT University
Vellore - India

Abstract: Each and every operating system has its own partition utility like parted for Unix based operating systems, disk management utility for Windows operating system and parted Util for ESX operating system. A partition utility for a particular operating system can able to detect only certain disk labels of other operating systems [1]. We need to develop a Utility by named partition table analyzer utility which has capability of analyzing a partition table if there is any partition table already exist on the disk .This utility is used to avoid data corruption in cloud environment.

Keywords: Cloud Operating System, Disk, Partition table, Partition Utility.

I. INTRODUCTION

To organize user's data more effectively all operating systems allow users to divide a hard disk into multiple partitions in order, making one physical hard disk is a defined storage space on a hard drive into several smaller logical hard disks. Each and every hard disk partition stores many file systems, in turn data is stored in each file system. All the information about partitions is used to store in the Partition table. The beginning sectors in a hard disk have a reserved area to store Partition table [2]. In creating, destroying, resizing, copying and checking partitions and file systems on them we use partition utility.

II. MOTIVATION

The Cloud operating system manages several storage resources and schedules the computing resources in a cluster of servers span across multiple data centers at geographically distributed locations [4][6][7]. When a system with particular operating system in cloud demands a storage resource for cloud, suppose cloud serves the storage resource which has been already a partition table with a disk label to the requested system [5]. If the requested system's partition utility cannot recognize the already existing partition table, it may lead to data corruption.

So there is a need to develop a Utility by named partition table analyzer utility which has capability of analyzing any partition table, if it already exists on the disk, to avoid data corruption in cloud environment. To avoid data corruption we have to develop a partition analyzer utility to which we provide Disk with certain disk-label and partition table and results in displaying Disk-label and contents in partition table.

III. Architecture for Partition Table Generator Utility

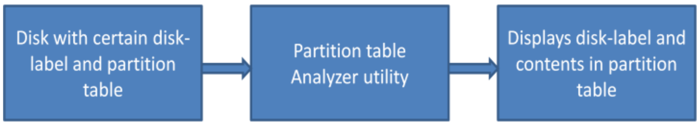

Fig. 1 Partition Table Generator Utility Architecture

IV. GUID Partition Table (GTP) Analyzer Algorithm

Algorithm for analyzing Guid Partition Table

Input: Disk-name with Guid Partition table
Output: Analyzing Guid Partition table

```
Algorithm for analyzing Guid Partition table
Begin
   Filedescriptor1←open(Disk-name, O_RDWR)
   read(Filedescriptor1, buffer, 34 * 512)
   sudoMbr←(Partition_MBR *) (buffer + 446)
   print  sudoMbr[0].type
   print sudoMbr[1].type
   print sudoMbr[2].type
   print sudoMbr[3].type
   gptHeader ← (Partition_GptHeader *) (buffer + 512)
   signature←0x5452415020494645
   if gptHeader→signature equals signature
      print valid Signature
   else  print invalid Signature
   revision←0x00010000
   if gptHeader→revision equals revision
      print valid Revision
   else
      print invalid Revision
   headerSize←0x5C
   if gptHeader→ headerSize equals headerSize
      print valid HeaderSize
   else
      print invalid HeaderSize
reserved1←0x0
   if gptHeader→ reserved1 equals reserved1
      print valid Reserved1
   else
      print invalid Reserved1
   myLba ←0x1
   if gptHeader→myLba equals myLba
      print valid MyLba Address
   else
      print invalid MyLba Address
   alternateLba←( gptHeader→lastUsableLba+33)
   if gptHeader→alternateLba equals alternateLba
      print valid AlternateLba Address
   else
      print invalid AlternateLba Address
   firstUsableLba ←0x22
   if gptHeader→firstUsableLba equals firstUsableLba
      print valid  FirstUsableLba Address
   else
      print invalid FirstUsableLba Address
   partitionEntryLba←0x2
   if gptHeader→partitionEntryLba equals partitionEntryLba
      print valid PartitionEntryLba Address
```

```
else
   print invalid PartitionEntryLba Address
numberOfPartitionEntries←0x80
if gptHeader→numberOfPartitionEntries
          equals numberOfPartitionEntries
   print valid Number Of Partition Entries
else
   print invalid Number Of Partition Entries
sizeofPartitionEntry←0x80
if gptHeader→sizeofPartitionEntry
         equals sizeofPartitionEntry
   print valid Size of Partition Entry
else
   print invalid Size of Partition Entry
temp←gptHeader→headerCrc32
gptHeader→headerCrc32=0x0
calCrc←efi_crc32(gptHeader, gptHeader→headerSize)
if calCrc equals temp
   print valid HeaderCrc32 in Primary Partition table header
else
   print invalid HeaderCrc32 in Primary Partition table header
close(Filedescriptor1)
Filedescriptor2←open(Disk-name, O_RDWR)
lseek(Filedescriptor2,
      512*gptHeader→alternateLba, SEEK_SET)
read(Filedescriptor2, buffer1, 512)
gptHeader1←(Secondary_Partition_GptHeader *)(buffer1)
signature←0x5452415020494645
if gptHeader1→signature equals signature
   print valid Signature
else
   print invalid Signature
revision←0x00010000
if gptHeader1→revision equals revision
   print valid Revision
else
   print invalid Revision
headerSize←0x5C
if gptHeader1→ headerSize equals headerSize
   print valid HeaderSize
else
   print invalid HeaderSize
reserved1←0x0
if gptHeader1→ reserved1 equals reserved1
   print valid Reserved1
else
   print invalid Reserved1
temp←gptHeader→lastUsableLba
if gptHeader1→ lastUsableLba equals temp
   print valid LastUsableLba Address in
               Secondary Partition header
else
   print invalid LastUsableLba Address in
                  Secondary Partition header
alternateLba ←0x1
if gptHeader1→alternateLba equals alternateLba
   print valid AlternateLba Address in
             Secondary Partition header
else
   print invalid AlternateLba Address in
                Secondary Partition header
myLba←( gptHeader1→lastUsableLba+33)
```

```
    if gptHeader1→myLba equals myLba
       print valid MyLba Address in Secondary Partition header
    else
       print invalid MyLba Address in Secondary Partition header
    firstUsableLba ←0x22
    if gptHeader1→firstUsableLba equals firstUsableLba
       print valid  FirstUsableLba Address
    else
       print invalid FirstUsableLba Address
    partitionEntryLba←0x2
    if gptHeader1→partitionEntryLba equals partitionEntryLba
       print valid PartitionEntryLba Address
    else
       print invalid PartitionEntryLba Address
    numberOfPartitionEntries←0x80
    if gptHeader1→numberOfPartitionEntries
             equals numberOfPartitionEntries
       print valid Number Of Partition Entries
    else
       print invalid Number Of Partition Entries
    sizeofPartitionEntry←0x80
    if gptHeader1→sizeofPartitionEntry equals sizeofPartitionEntry
       print valid Size of Partition Entry
    else
       print invalid Size of Partition Entry
    temp←gptHeader1→headerCrc32
    gptHeader1→headerCrc32=0x0
    calCrc←efi_crc32(gptHeader1, gptHeader1→headerSize)
    if calCrc equals temp
       print valid HeaderCrc32 in Secondary Partition header
    else
       print invalid HeaderCrc32 in Secondary Partition header
  close(Filedescriptor2)
End
```

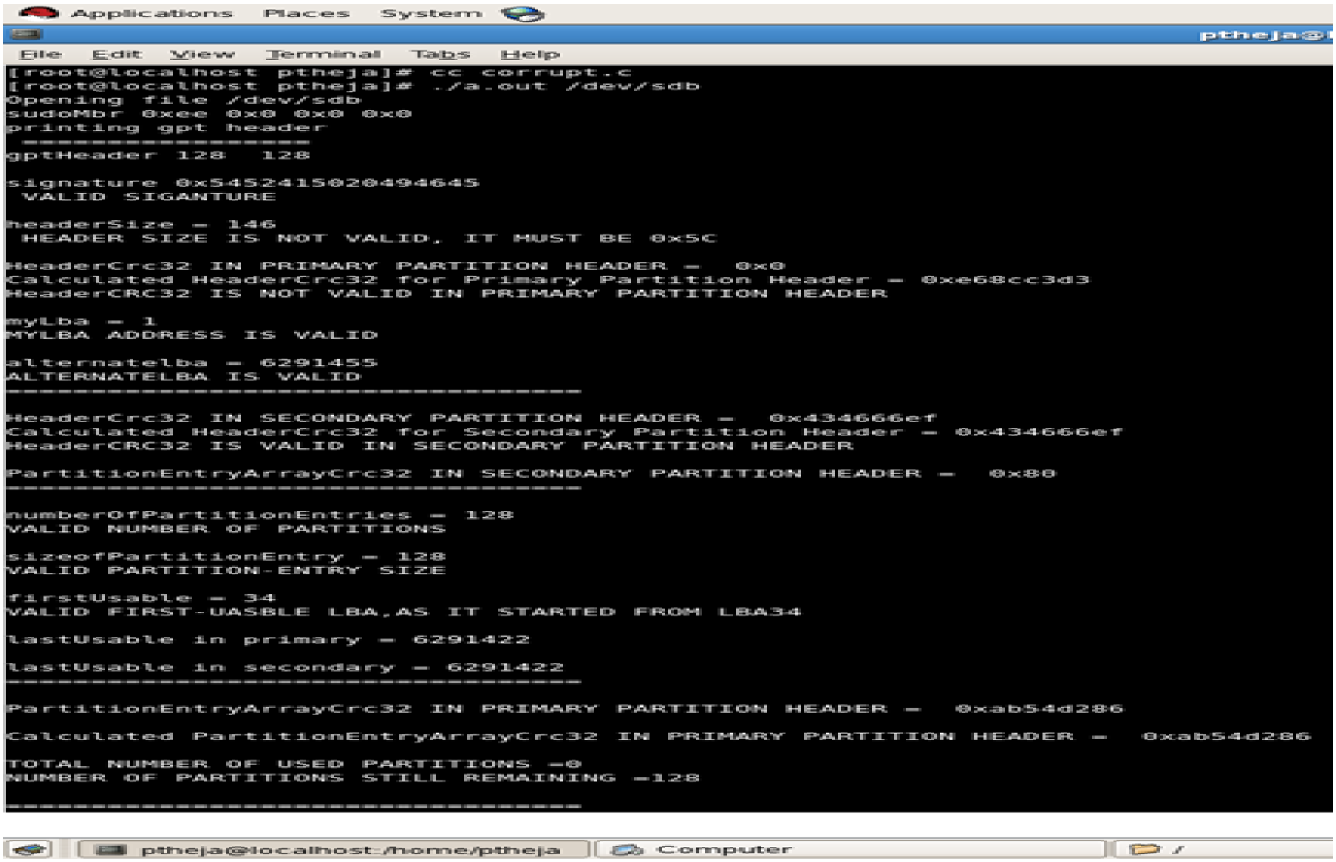

Fig 2: Screen shot for creating GUID Partition Table

TABLE I : GUID Partition Table Contents

Mnemonic	Byte Offset	Byte Length	Description
Signature	End	Last	Identifies EFI-compatible partition table header. This value must contain the string "EFI PART", 0x5452415020494645.
Revision	8	4	The specification revision number that this header complies to. For version 1.0 of the specification the correct value is 0x00010000.
HeaderSize	12	4	Size in bytes of the GUID Partition Table Header.
HeaderCRC32	16	4	CRC32 checksum for the GUID Partition Table Header structure. The ranged defined by HeaderSize is "check-summed".
Reserved	20	4	Must be Zero.
MyLBA	24	8	The LBA that contains this data structure.
AlternateLBA	32	8	LBA address of the alternate GUID Partition Table Header.
FirstUsableLBA	40	8	The first usable logical block that may be contained in a GUID Partition Entry.
LastUsableLBA	48	8	The last usable logical block that may be contained in a GUID Partition Entry.
DiskGUID	56	16	GUID that can be used to uniquely identify the disk.
PartitionEntry LBA	72	8	The starting LBA of the GUID Partition Entry array
NumberOfPartitionEntries	80	4	The number of Partition Entries in the GUID Partition Entry array.
SizeOfPartitionEntry	84	4	The size, in bytes, of each the GUID Partition Entry structures in the GUID Partition Entry array. Must be a multiple of 8.
PartitionEntry ArrayCRC32	88	4	The CRC32 of the GUID Partition Entry array. Starts at Partition Entry LBA and is (NumberOfPartitionEntries)* (SizeOfPartitionEntry in byte length.)
Reserved	92	92	The rest of the block is reserved by EFI and must be zero.

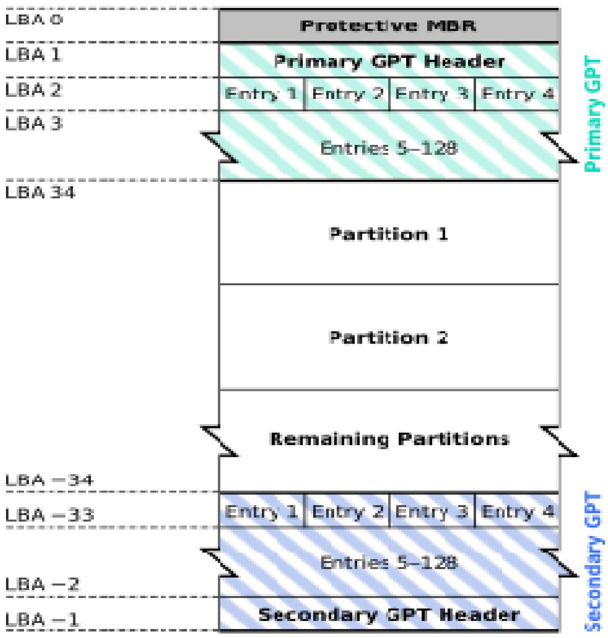

Fig 3: GUID Partition Table (GPT) Scheme

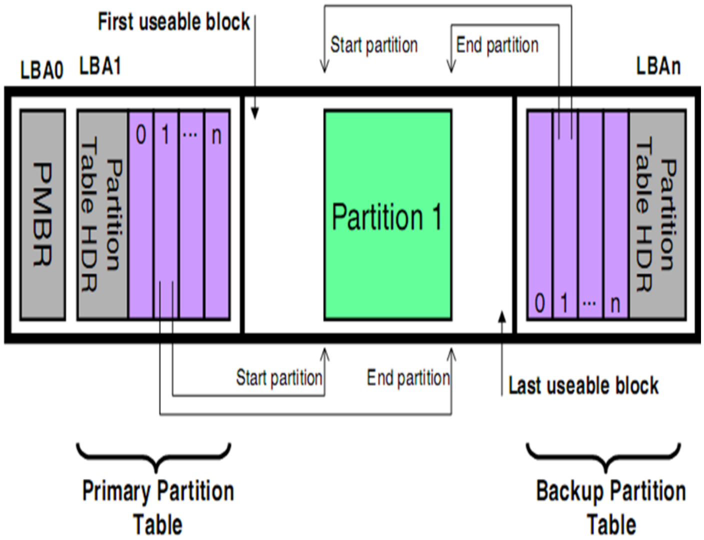

Fig 4: GUID Partition Table (GPT) Scheme [3]

TABLE II : GUID Partition Entry Contents [3]

Mnemonic	**Byte Offset**	**Byte Length**	**Description**
Partition Type Guid	0	16	Unique id that defines the purpose and type of this Partition. A value of zero defines that this partition record is not being used.
Unique Partition Guid	16	16	Guid that is unique for every partition record.
StartingLBA	32	16	Starting LBA of the partition defined by this record.
EndingLBA	40	8	Ending LBA of the partition defined by this record.
Attributes	48	8	Attribute bits, all bits reserved by EFI.
Partition Name	56	72	Unicode string.

V. MASTER BOOT RECORD PARTITION (MBR) ANALYZER ALGORITHM

Algorithm for analyzing Master Boot Record Partition Table

```
Input: Disk-name with MBR Partition table
Output: Analyzing MBR Partition table

Algorithm for analyzing MBR Partition table
Begin
    Char buffer[512]
    Filedescriptor1←open(Disk-name, O_RDWR)
    MSDOS_LABEL_MAGIC←0xAA55
    label←(unsigned short *)(buffer +446)
    if *label equals MSDOS_LABEL_MAGIC
         print MSDOS label found
    else
         print MSDOS label not found
    SCSI_PTABLE_SECTOR_OFFSET
    p←(sector) ((Partition_MBR *)(((uint8 *)(sector))
      + SCSI_PTABLE_SECTOR_OFFSET))
    for i=0 to 4
    if(p→boot_ind==0x80) then
         print STATUS IS BOOTABLE(ACTIVE)
              elseif(p→boot_ind==0x00) then
           print STATUS IS NON-BOOTABLE(INACTIVE)
      else
           print STATUS IS INVALID
          end if then
```



```
        if(p→type==0x0f)
                print PARTITION TYPE IS EXTENDED
                      PARTITION WITH LBA
         elseif(p→type==0x05)
                print PARTITION TYPE IS EXTENDED
                PARTITION WITH CHS ADDRESS
         elseif(p→type==0x00)
            print EMPTY PARTITION
         else
             print PARTITION TYPE IS PRIMARY PARTITION
  end if then
  FirstSector←p→firstSector
  NumSectors←p→ numSectors
   StartingLba←p→ firstSector
    EndingLba←(p→firstSector+ p→ numSectors)
    print FirstSector, NumSectors, StartingLba, EndingLba
    p++
      end i-loop
    close(Filedescriptor1)
 End
```

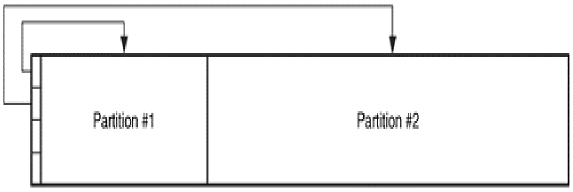

Fig 5: A Basic DOS disk with two partitions and one MBR

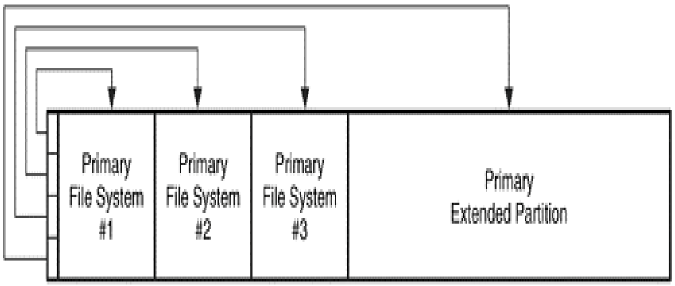

Fig 6: A DOS disk with three primary file system partitions and one primary secondary partition.

TABLE III: MBR Partition Table Contents [3]

Mnemonic	**Byte Offset**	**Byte Length**	**Description**
BootCode	0	440	Code used on legacy Intel architecture system to select a partition record and load the first block (sector) of the partition pointed to by the partition record. This code is not executed on EFI systems.
uniqueMBR Signature	440	4	Unique Disk Signature, this is an optional feature and not on all hard drives. This value is always written by the OS and is never written by EFI firmware.
Unknown	444	2	Unknown
Partition Record	446	16*4	Array of four MBR partition records.
Signature	510	2	Must be 0xaa55.

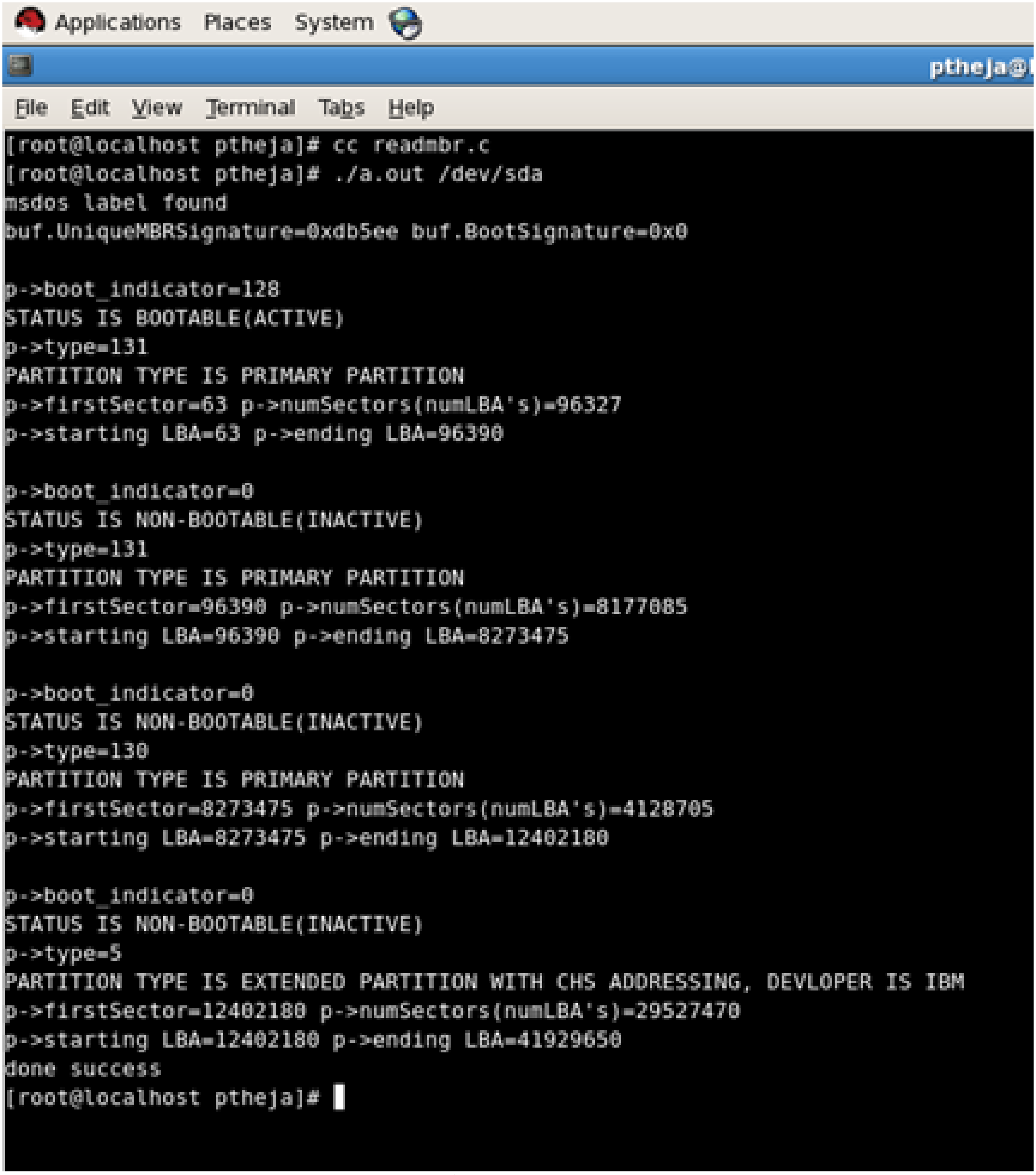

Fig 7: Screen shot for creating MBR Partition Table

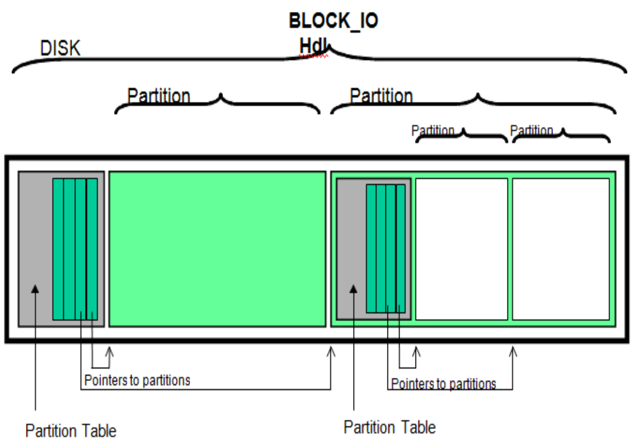

Fig 8: Nesting of Legacy MBR Partition Table [3]

VI. MAC PARTITION TABLE (MAC) ANALYZER ALGORITHM

Algorithm for analyzing MAC Partition Table

Input: Disk-name with MAC Partition table
Output: Analyzing MAC Partition table

```
Algorithm for Analyzing MAC Partition table
Begin
    Char buffer[4096]
    Filedescriptor1←open(Disk-name, O_RDWR)
       q←(MacRawDisk *)(buffer +0)
       print q→signature, q→block_size, q→block_count
    for i=512 to 8*512
       p←(MacRawPartition *)(buffer +i)
if(p→ signature==0x4d50) then
      print p→signature, p→res1, p→map_count
      print p→start_block, p→block_count
      for i=0 to 31
           print p→name[i], p→type[i]
      print p→data_start, p→data_count
      if(p→status== 0x00000001)
              print 'Entry is valid'
           elseif(p→status== 0x00000002)
              print 'Entry is allocated'
           elseif(p→status== 0x00000004)
              print 'Entry in use'
           elseif(p→status== 0x00000008)
              print 'Entry contains boot information'
           elseif(p→status== 0x00000010)
              print 'Partition is readable'
           elseif(p→status== 0x00000020)
              print 'Partition is writable'
           elseif(p→status== 0x00000040)
              print 'Boot code is position independent'
           elseif(p→status== 0x00000100)
              print 'Partition contains chain-compatible driver'
           elseif(p→status== 0x00000200)
              print 'Partition contains a real driver'
           elseif(p→status== 0x00000400)
              print 'Partition contains a chain driver'
           elseif(p→status== 0x40000000)
              print 'Automatically mount at startup'
elseif(p→status== 0x80000000)
              print 'The startup partition'
           else
               print 'Reserved'
        print p→boot_start,
              p→boot_count, p→boot_load,boot_load2
        print p→boot_entry, p→boot_entry2, p→boot_cksum
        for i=0 to 16
           print  p→processor[i]
        print p→driver_sig
       else
        print 'Analysis is finished'
       i←i+512
 end i-loop
 close(Filedescriptor1)
End
```

TABLE IV: Apple Partition Table Contents [2]

Byte Range	Description	Essential
0-1	Signature Value (0x504D)	NO
2-3	Reserved	NO
4-7	Total Number of partitions	YES
8-11	Starting sector of partition	YES
12-15	Size of partition in sectors	YES
16-47	Name of partition in ASCII	NO
48-79	Type of partition in ASCII	NO
80-83	Starting sector of data area in partition	NO
84-87	Size of data area in sectors	NO
88-91	Status of partition	NO
92-95	Starting sector of boot code	NO
96-99	Size of boot code in sectors	NO
100-103	Address of boot loader code	NO
104-107	Reserved	NO
108-111	Boot code entry point	NO

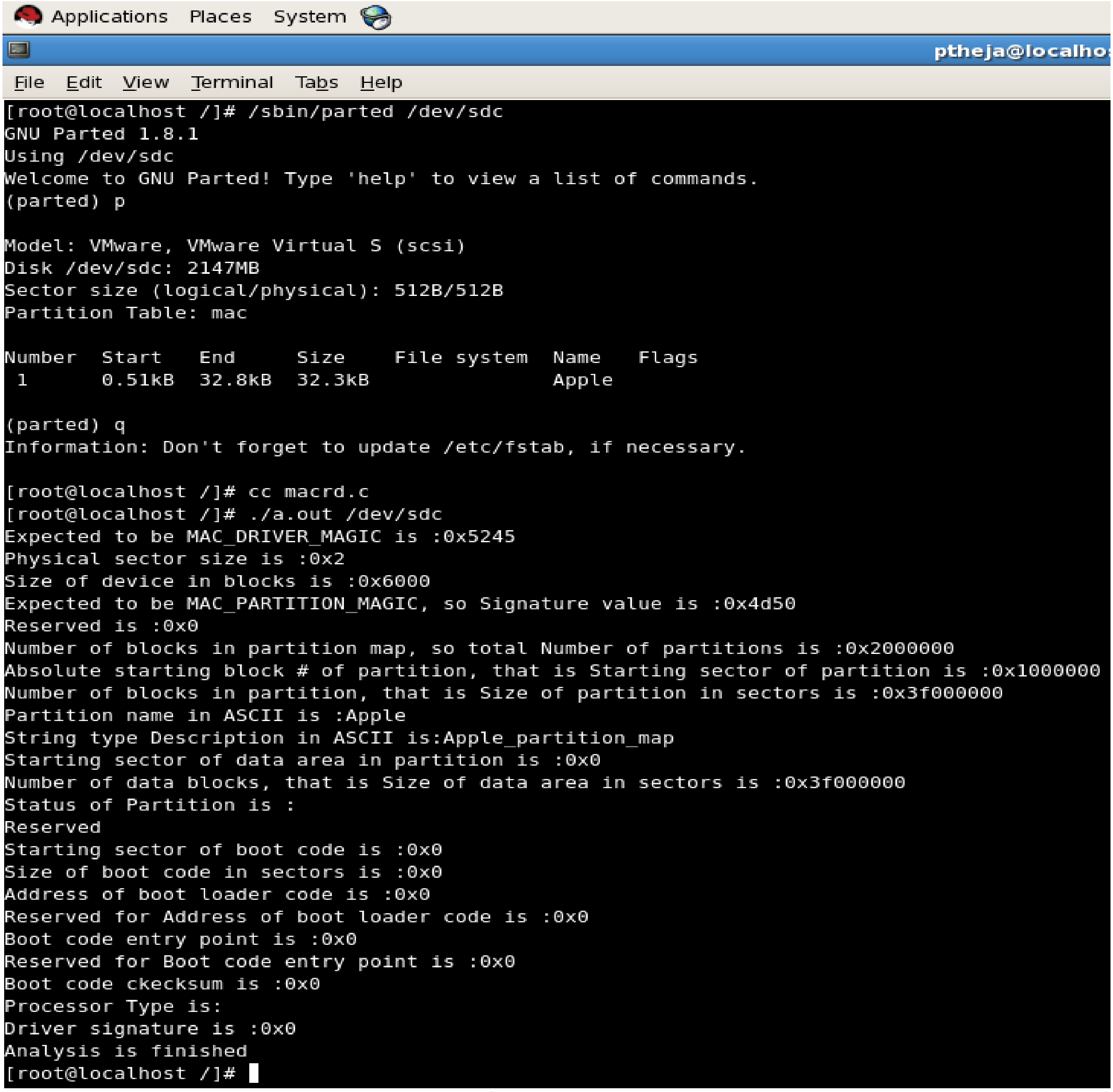

Fig 9: Screen shot for creating MAC Partition Table

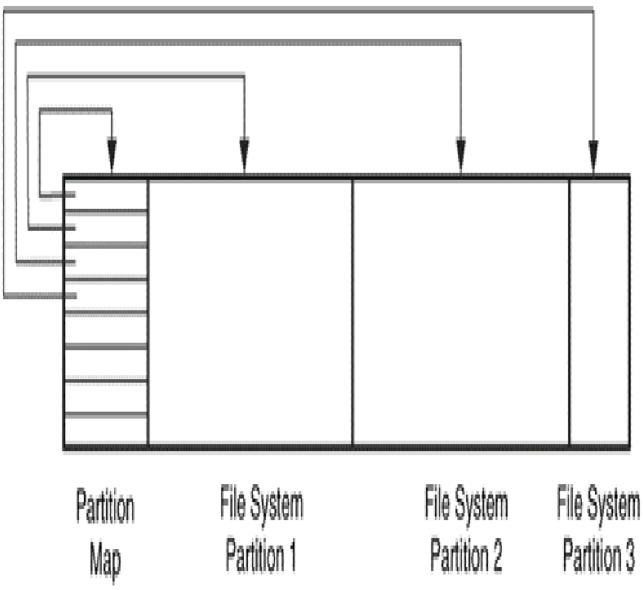

Fig 10: An Apple disk with one partition map partition and three file system partitions.

VII. SUN PARTITION TABLE (SUN) ANALYZER ALGORITHM

Algorithm for analyzing SUN Partition Table

Input: Disk-name with SUN Partition table
Output: Analyzing SUN Partition table

```
Algorithm for analyzing SUN Partition table
Begin
 Char buffer[512]
 Filedescriptor1←open(Disk-name, O_RDWR)
 nheads← (unsigned short *)(buf+437)
 lseek(Filedescriptor1,437,SEEK_SET)
 print *nheads
 ntracks← (unsigned short *)(buf+439)
 lseek(Filedescriptor1,439,SEEK_SET)
 print *ntracks
 label1← (unsigned short *)(buf+508)
 lseek(Filedescriptor1,508,SEEK_SET)
 print*label1
 disk_speed← (unsigned short *)(buf+420)
 lseek(Filedescriptor1,420,SEEK_SET)
 print *disk_speed
 phy_cylinders← (unsigned short *)(buf+422)
       lseek(Filedescriptor1,422,SEEK_SET)
 print *phy_cylinders
 version← (unsigned short *)(buf+431)
 lseek(Filedescriptor1,431,SEEK_SET)
 print *version
 phy_cylinders1← (unsigned short *)(buf+432)
 lseek(Filedescriptor1,432,SEEK_SET)
 print *phy_cylinders1
 disk_size← (unsigned long *)(buf+464)
 lseek(Filedescriptor1,464,SEEK_SET)
 print *disk_size
 checksum← (unsigned short *)(buf+510)
 lseek(Filedescriptor1,510,SEEK_SET)
 print *checksum
 close(Filedescriptor1)
End
```

TABLE V: Data Structure for the SUN SPARC Disk label[2]

Byte Range	Description	Essential
0-127	ASCII Label	NO
128-261	Sparc VTOC	YES
262-263	Sectors to skip, writing	NO
264-265	Sectors to skip, reading	NO
266-419	Size of partition in sectors	NO
420-421	Disk speed	NO
422-423	Number of physical cylinders	NO
424-425	Alternates per cylinder	NO
426-429	Reserved	NO
430-431	Interleave	NO
432-433	Number of data cylinders	NO
434-435	Number of alternate cylinders	NO
436-437	Number of heads	YES

438-439	Number of sectors per track	YES
440-443	Reserved	NO
444-451	Partition #1 disk map	YES
452-459	Partition #2 disk map	YES
460-467	Partition #3 disk map	YES
468-475	Partition #4 disk map	YES
476-483	Partition #5 disk map	YES
484-491	Partition #6 disk map	YES
492-499	Partition #7 disk map	YES
500-507	Partition #8 disk map	YES
508-509	Signature Value (0xDABE)	NO
510-511	Checksum	NO

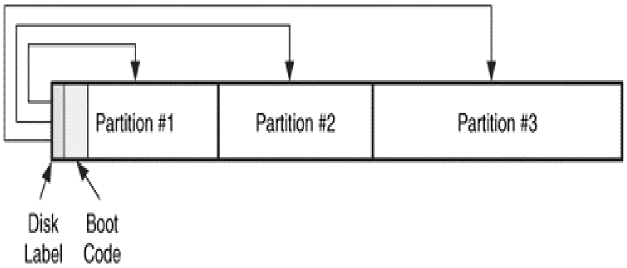

Fig 11: An SUN disk with three dos partitions, the final one contains a disk label and three SUN partitions.

TABLE VI: Data Structure for the VTOC in SUN SPARC Disk label [2].

Byte Range	Description	Essential
0-3	Version (0x01)	NO
4-11	Volume Name	NO
12-13	Number of Partitions	YES
14-15	Partition #1 type	NO
16-17	Partition #1 flags	NO
18-19	Partition #2 type	NO
20-21	Partition #2 flags	NO
22-23	Partition #3 type	NO
24-25	Partition #3 flags	NO
26-27	Partition #4 type	NO
28-29	Partition #4 flags	NO
30-31	Partition #5 type	NO
32-33	Partition #5 flags	NO
34-35	Partition #6 type	NO
36-37	Partition #6 flags	NO
38-39	Partition #7 type	NO
40-41	Partition #7 flags	NO
42-43	Partition #8 type	NO
44-45	Partition #8 flags	NO
46-57	Boot info	NO
58-59	Reserved	NO

60-63	Signature Value -0x600DDEEE	NO
64-101	Reserved	NO
102-105	Partition #1 timestamp	NO
106-109	Partition #2 timestamp	NO
110-113	Partition #3 timestamp	NO
114-117	Partition #4 timestamp	NO
118-121	Partition #5 timestamp	NO
122-125	Partition #6 timestamp	NO
126-129	Partition #7 timestamp	NO
130-133	Partition #8 timestamp	NO

TABLE VII: Data Structure for the SUN SPARC Disk label Disk Map [2]

Byte Range	Description	Essential
0-3	Starting Cylinder	YES
4-7	Size	YES

VIII. AMIGA PARTITION TABLE (AMIGA) ANALYZER ALGORITHM

Algorithm for analyzing AMIGA Partition Table

Input: Disk-name with AMIGA Partition table
Output: Analyzing AMIGA Partition table

```
Algorithm for analyzing AMIGA Partition table
Begin
    Char buffer[4096]
     Filedescriptor1←open(Disk-name, O_RDWR)
     p←(RigidDiskBlock *)(buffer +1024)
     if(p→rdb_ID==0x4b534452)
          print ‘Amiga Label is 0x4b534452’
       else
          print ‘Amiga Label is not Found’
     print p→ rdb_SummedLongs, p→ rdb_ChkSum,
     print p→ rdb_HostID, p→ rdb_BlockBytes
     print p→ rdb_Flags
    close(Filedescriptor1)
End
```

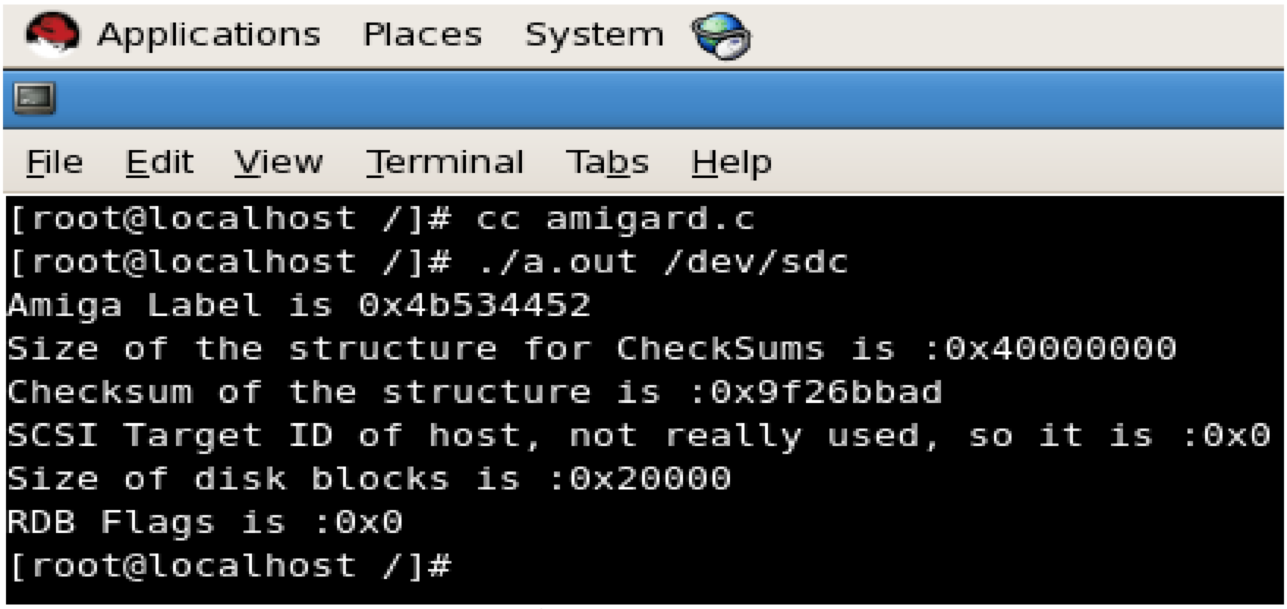

Fig 12: Screen shot for creating AMIGA Partition Table

IX. BSD PARTITION TABLE (BSD) ANALYZER ALGORITHM

Algorithm for analyzing BSD Partition Table

```
Input: Disk-name with BSD Partition table
Output: Analyzing BSD Partition table

Algorithm for analyzing BSD Partition table
Begin
    Char buffer[512]
    Filedescriptor1←open(Disk-name, O_RDWR)
       p←(BSDRawLabel *)(buffer +64)
       if(p→d_magic== BSD_DISKMAGIC)
           print 'BSD label is Found that is : 0x82564557'
         else
           print 'BSD label is not Found'
       if(p→d_type==0x1)
           print 'Drive type is :BSD_DTYPE_SMD
                 that is SMD<XSMD'
         elseif(p→d_type==0x2)
           print 'Drive type is :BSD_DTYPE_MSCP'
         elseif(p→d_type==0x3)
           print 'Drive type is :BSD_DTYPE_DEC
                 that is other DEC (rk,rl)'
         elseif(p→d_type==0x4)
           print 'Drive type is :BSD_DTYPE_SCSI'
         elseif(p→d_type==0x5)
           print 'Drive type is :BSD_DTYPE_ESDI
                 that is ESDI interface'
         elseif(p→d_type==0x6)
           print 'Drive type is :BSD_DTYPE_ST506'
         elseif(p→d_type==0x7)
           print 'Drive type is :BSD_DTYPE_HPIB
                 that is CS/80 on HP-IB'
         elseif(p→d_type==0x8)
           print 'Drive type is :BSD_DTYPE_HPFL
                 that is HP Fiber-link'
         elseif(p→d_type==0xa)
           print 'Drive type is :BSD_DTYPE_FLOPPY
                 that is floppy'
         else  print 'bsd dtype is not found.'
       print p→d_subtype, p→d_secsize,
            p→d_nsectors, p→d_ntracks
      print p→d_ncylinders, p→d_secpercyl, p→d_secperunit
      print p→d_sparespercyl, p→d_acylinders,
                  p→d_rpm, p→d_interleave
            print p→d_trackskew, p→d_cylskew,
                  p→d_headswitch, p→d_trkseek
            print p→d_flags, p→d_sparespertrack
         for i=0 to 5
            print p→d_driverdata[i]
         print p→d_magic2, p→d_checksum, p→d_npartitions
         print p→d_bbsize, p→d_sbsize
         close(Filedescriptor1)
      End
```

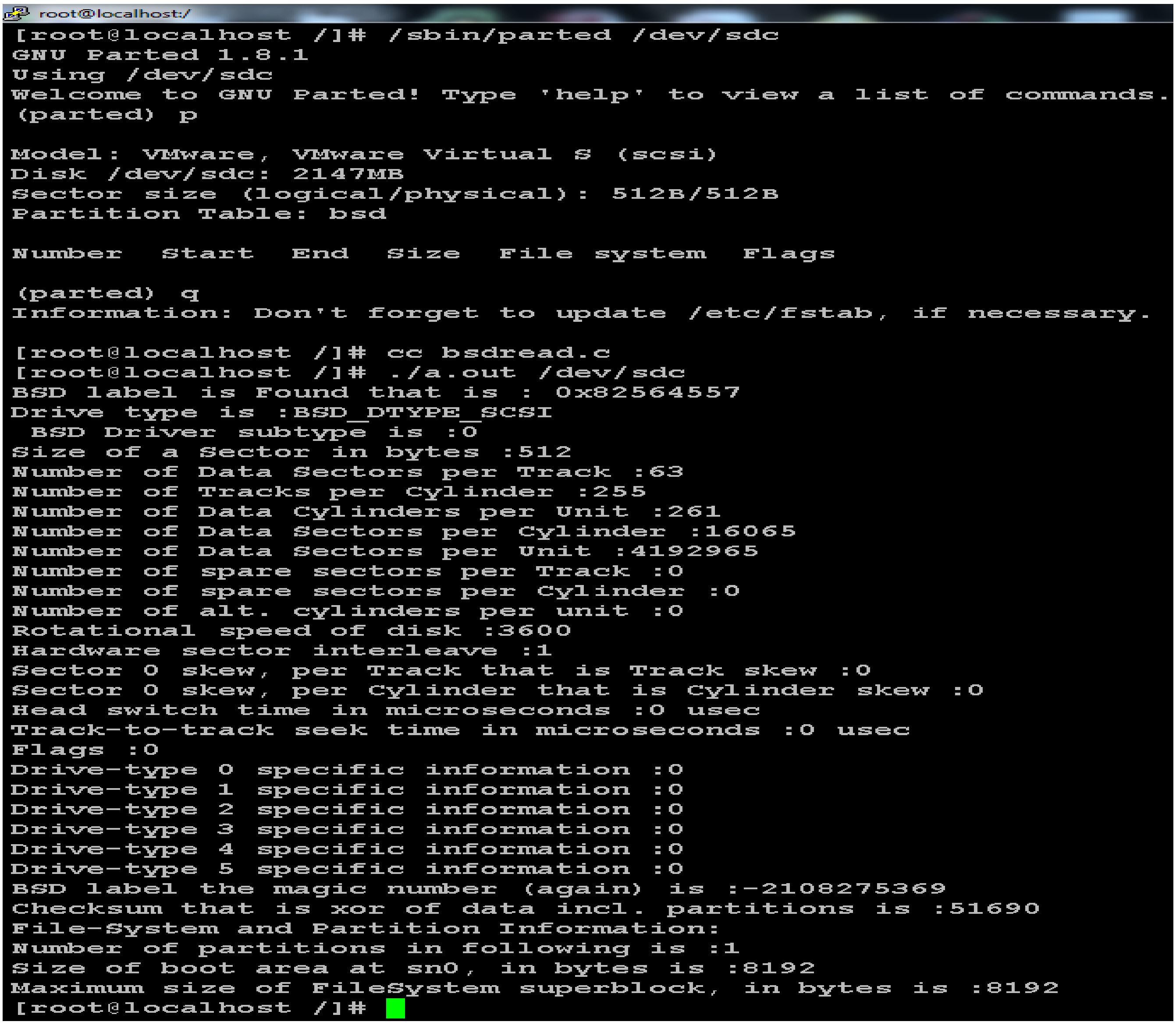

Fig 13: Screen shot for creating BSD Partition Table

X. Conclusion

We clearly stated algorithms of how to detect and analyze different major partition tables with disk labels. We have performed analysis on all major partition tables and succeeded in detecting and analyzing them. In future enhancement, the analysis will be extended on remaining partition tables and make this utility strong enough of detecting any specified partition table. The conclusion of the paper is to avoid data corruption in cloud environment by using partition analyzer utility.

References

1. Perla Ravi Theja, Govinda K and P Swarnalatha. Article: "Partition Table Generator for Cloud OS", International Journal of Computer Applications, 41(16):1-15, March 2012. Published by Foundation of Computer Science, New York, USA. Available: http://research.ijcaonline.org/volume41/number16/pxc3877750.pdf
2. Brain Carrier. (2005) "File System Forensic Analysis", ISBN:0-32-126817-2 , pp-66-109.
3. Intel. (2000) "Extensible Firmware Interface Specification", Version 1.02, pp- 305-316.
4. A+ Guide to PC Hardware Maintenance and Repair, 1st ed., Michael Graves, Cengage Learning, United States, 2004.
5. The Multi-Boot Configuration Handbook, 1st ed., Roderick W. Smith, Que Publishing, United States, 2000.
6. Roderick W. Smith, aka rodsbooks.com, (2011) [online]. Oberlin College. Available: http://rodsbooks.com/gdisk. [Accessed 10 Jan 2011].

AUTHOR(S) PROFILE

Perla Ravi Theja, received the Master of Technology in Computer Science and Engineering from School of Computer Science and Engineering (SCSE), VIT University, Vellore in 2012.
He is currently working as a Programmer Analyst for the Cognizant Technology Solutions and pursuing PhD degree in Computer Science and Engineering. During his Masters, he spent ten months as an intern at VMware, Bangalore. His research interests include cloud, storage area networks, partition tables, iscsi, fiber channel, cloud security and virtualization. He has published several papers in international journals and conferences related to his research topics.
Mr. Perla Ravi Theja is a member of the IEEE Computer Society. He has awarded best student award in his school during his masters. He has published "Partition Table Generator for Cloud OS", in International Journal of Computer Applications (IJCA), 41(16):1-15, March 2012. http://research.ijcaonline.org/volume41/number16/pxc3877750.pdf.

Varun Manchikalapudi, received Master of Technology in Information Technology, Networking Specialization from School of Information Technology and Engineering (SITE), VIT University, Vellore in 2010.
He is working as Assistant Professor in Department of Information Technology, V R Siddhartha Engineering College, Vijayawada since 2010. He has an industry experience, Worked as Software Intern in Honeywell Technology Solutions Lab, Bangalore. He has published two research papers on wireless networks in IEEExplore digital library. He has published a paper on image processing in an International Journal (IJCST) in 2011.

Dr. SK. Khadar Babu, working as Senior Assistant Professor in VIT University, Vellore. He is working under statistics and operations research division in school of Advanced Sciences. His area of interests include computer networks, cloud computing, Digital image processing, ARIMA models, Statistical quality control, Fuzzy programming techniques.