omniplay

COLLABORATORS			
	TITLE : omniplay		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 2, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

omniplay

1.1 oplay.guide

```
                OmniPlay
version 1.231

A Multiple-Format Sound Player for the Amiga


Contents:


                Background
                   How it happened

                Syntax
                   How to use OmniPlay

                Examples
                   Example commands

                Miscellaneous
                   Notes on specific features

                Formats
                   What the formats are like

                Legal Stuff
                   Legal Stuff

                Change History
                 When things happened

                Known Bugs
                   Known Bugs in OmniPlay

                Credits
                   Who I thank

                Author
```

Who I am

1.2 background

Background:

In the beginning, the Amiga could play sounds. Few other machines could go beyond beeps, bells, and simple waveform descriptions. In the last few years, however, many manufacturers have been developing sampled sound playback hardware for their computers, and with almost every new sound generation system, there's a new sound file format. While it's much easier now to find interesting samples, it's much harder to listen to them. Tools are constantly developed on every platform to read one format and save it as something a local player program recognizes. There have been many primitive converters for many types of machine, usually converting only one type of file to one other type of file. It is a mess.

A couple of years ago, Lance Norskog began work on the Sound Exchange, a program and linker library kit that can both read and write almost any sound file type, and handle data flow between them. It was originally created for Unix, but recently it's been ported to a number of other platforms, as well. Finally, it was possible to deal with almost any sound in existence. But you still had to figure what a file was, then convert it to something you know, and find a good player for that format.

How OmniPlay came to be...

I've heard lots of requests lately on
Usenet
for Amiga
programs that will play [insert format here] sound files. The standard answer has always been "Use MegaMondo Bigtime Sound Converter (
MBSC
) (sorry, no cute acronym) to change your
SPIFF
file to
IFF
, then play it with Killer Buttkick Sound Player (
KBSP
, version 2.1 only!)."

Previously, I had ported
SOX
to the Amiga as a temporary solution to the myriad of unplayable sound formats, but almost everyone finds that while SOX is quite flexible, it's

just plain slow. Mike Cramer has been steadily developing a fast Amiga-only sound file converter, but the fact remains that it's a converter and not a player. This is fine, but sometimes you just want to play a file without changing it. Besides, some formats (such as
u-law
, 16-bit
WAV
, 16-bit

AIFF
, and 16-bit
.snd
) store data that would be lost forever by conversion, but merely playing the files only loses the data for the duration of playback. Finally I started thinking: "I've got to write most of the essential bits anyway for this other program I'm doing, so I might as well code the file reader and playroutine now to satisfy everyone." (By now, of course, the file reader and playroutine comprise a tool useful in itself, and development should go on past the possible completion of the "other program".)

...and what it is:

Well, here you go. OmniPlay is an Amiga sound player that reads and plays
IFF-8SVX
,
IFF-AIFF
, Sun
.au
, NeXT
.snd
(u-law, linear-8, linear-16),
WAV
,
VOC
,
signed raw
,

unsigned raw
, and raw
u-law
sound formats. OmniPlay determines the sound type on its own, so you needn't tell it what format something is (although you can if you wish), and it reads all relevant playback information from headered formats (although you can override these on the command line). Additionally, optional properties of a formatted sound (e.g., IFF's NAME, AUTH, etc.) are (optionally) printed to the output stream (usually console) whenever encountered in a file. OmniPlay also plays
PowerPacked

```
                sounds, as long
  as they decompress to one of the above formats.  Furthermore,
  OmniPlay is as fast as or faster than the average sound
  converter, and can parse a sound file quickly enough to keep
  up with the audio device.  Unless you need IFF sounds for
  another application, there's no real need to convert sounds.
```

1.3 syntax

```
                Command Syntax:

> oplay [-r rate] [-v volume] [-b buffersize] [-c l|r|e] [-o 0|1]
  [-p pri] [-w ticks] [-f a|l|s|u] [-s 0|1] [-d PAL|NTSC]
  [-g 0|1] [-m 0|1] [-a factor] [-W filename] [-i 0|1]
  [-O filespec] [-R cycles] [--file] file [...]


                 -r
                :     Override default playback rate with rate.

                 -v
                :     Override default playback volume with volume.

                 -b
                :     Set size of one buffer (both are the same size).

                 -o
                :     One-shot: load it all to RAM before playing.

                 -c
                :     Set channel for playback.  Left, Right, or Either.

                 -p
                :     Set oplay's task priority.

                 -f
                :     Force playback as Auto, u-Law, Signed raw, or Unsigned raw

                 -w
                :     Pause/Wait for <ticks> * 1/50 second.

                 -s
                :     Show (1) or don't show (0) optional properties.

                 -d
                :     Set electrical standard (PAL or NTSC).

                 -g
                :     Generate log tables for ulaw files.

                 -m
                :     Show u-law maximum encoded value.

                 -a
                :     Amplify by multiplying sound by an integer.
```

```
 -W
:     Write sound as an IFF to "filename".

 -i
:     Display more info on sound file.

 -O
:     A file name or window spec to receive output.

 -R
:     Repeat: play the following sounds <cycles> times.

 --
:     Prevent option parsing for this argument.
```

Option switches are case-sensitive. Option arguments might or might not be case-sensitive, depending on their use.

Option arguments are allowed, but not required, to be separated from the corresponding option switch by whitespace (either spaces or tabs).

Note that none of the options are necessary for playback, but they might help you get the best sound quality from a file.

1.4 -r

```
Option description:
  -r: Use this option to adjust the playback rate of a
    sound.  This overrides any definition stored in the
    header of a sound, or the default for raw sounds.
    Default: format-specified, or 10000 (8000 for ulaw,
    11395 for unsigned).
```

1.5 -v

```
Option description:
  -v: To control gain on a "hot" sound.  Range is 0 to 64.
    Note that this has nothing to do with sample
    distortion or peaking.
    Default: 64.
```

1.6 -b

```
                Option description:
  -b: To define how much data is read between buffer-
    loading.  OmniPlay allocates two buffers, each of the
    given byte size, for playback.  Default: 4000 bytes.
```

```
  You may experience some slowdown in 16-bit or u-law sounds on
  slow machines with small buffers.  This is because OmniPlay
  does some conversions during playback so as not to have to
  load everything in at once.  Using a buffer half as big as
  the sound will usually cure this completely; using a smaller,
  but still large, buffer will make it break less often.  The
  best solution is the
                -o
                option, which allocates a single
  buffer sized precisely to the length of the sound.
```

1.7 -o

```
Option description:
  -o: If you don't want OmniPlay to continuously read while
    playing, set this to 1.  OmniPlay will find the size
    of its input and adjust the buffer size so that it
    can read all of the file in one shot, rather than
    reading during playback.  Using '-o0' means that
    only one bufferfull is read at a time, so you get
    immediate playback and continuous disk access.  Use
    '-o1' to play from floppy, slow NFS volumes, or other
    slow devices without choppiness.  Naturally, you must
    have enough RAM to load the whole sound to do this.
    '-o1' also overrides any value you set for '-b'.
    Default: 0.

  When using the -o1 setting, only one buffer is allocated.
  This one buffer is set to the exact size of the sound file,
  and it is loaded in full before playback begins.  See the
  Double-buffering section to contrast this method with the
  default ('-o0') method.
```

1.8 -c

```
Option description:
  -c: Channel selection.  Use '-cl' for Left, '-cr' for
    Right, or '-ce' for Either.  "Either" means to take
    whatever channel is available first.  Default: e.
```

1.9 -p

```
Option description:
  -p: Adjust process priority.  This was for debugging,
    originally -- when an input handler hung, I wanted
    oplay not to hog the system while I figured out what
    it shouldn't have assumed about its input.  (This is
    no longer a problem; all handlers work now.)  You
    might use it, for example, to play a long sound in
    the background, but not to use CPU time that could be
```

```
    used by your PostScript interpreter, compiler,
    archiver, or any other CPU-intensive program.
    Default: -1.
```

1.10 -f

```
                Option description:
  -f: Suppose, just suppose, you have a WAV file, say, that
    OmniPlay refuses to read.  You can make it play it in
    8 bits with '-fu'.  For
                 AIFF
                ,
                 .au/.snd
                 (linear), or

                 8SVX
                , use '-fs'; for
                 WAV
                 &
                 VOC
                  use '-fu'; for

                 .au/.snd
                 (u-law), use '-fl'.  Like
                 -r
                 and
                 -v
                ,
    this option maintains control until disabled.  To
    disable, use '-fa' (for auto).  Default: a.


  When playing raw sounds, OmniPlay pauses for any number of
  nanoseconds while it figures out what sort of raw file this
  is.  This timeout can be avoided by specifying some raw
  format with this option.
```

1.11 -w

```
Option description:
  -w: You can cause simple scripting by making oplay pause
    for some amount of time before going on to the next
    sound.  Note that wait-times are measured in 1/50
    seconds for precision.  Default: 0.
```

1.12 -s

```
Option description:
  -s: Show properties.  If you don't care what program
    digitized the sound, or think that console output
```

```
    would interfere with whatever else is going on, you
    can use "-s0".  Default: 1 (property information on).
```

1.13 -d

```
Option description:
  -d: If you have an NTSC Whoppin' Agnus booted into PAL
    mode (or vice versa, for whataver reason), you
    should set this to what you really are.
    Default: whataver GfxBase says, or NTSC if GfxBase
    doesn't say 'PAL' or 'NTSC'.
```

1.14 -g

```
                Option description:
  -g:
                 U-law
                 files are 14-bit samples logarithmically
    encoded to 8 bits.  To play these files, OmniPlay
    must convert u-law encoding to 8-bit linear, which
    requires the generation of "log tables" which equate
    encoded values to linear values.  Computing this
    table can take some time, especially for large u-law
    files.  This option defines whether log tables are
    generated for the special characteristics of an
    individual u-law file ("-g1"), or computed for a
    generic u-law of full 14-bit resolution ("-g0").
    The latter is much faster, since it doesn't have to
    scan the file for a maximum encoding value, but can
    produce output that's significantly less audible than
    that produced with a customized log table.  Most of
    my u-law sounds, however, *do* use full resolution,
    so "-g0" is acceptable.  Default: 0.
```

1.15 -m

```
Option description:
  -m: If a u-law file plyed with "-g0" set isn't very loud,
    you might want to check its maximum encoding value.
    "-m1" causes OmniPlay to print this maximum; "-m0"
    suppresses it.  If the maximum is less than, say,
    20,000, you might get better sound quality by using
    "-g1" (at the expense of calculation time).
    Default: 0.
```

1.16 -a

```
Option description:
  -a: Many sounds are digitized at low gain, so that not
    only resolution (therefore quality) is lost, but the
    volume of the sound is quite low -- often completely
    inaudible.  Using this option, you can amplify the
    sound's volume (but not its quality) by any integral
    amount.  Default: 1 (= 100%).
```

1.17 -_w

```
                Option description:
  -W: Well, it was easy to do, so I did.  This option
    creates an
                 IFF-8SVX
                 file for the sound you're
    playing.  It duplicates any optional properties from

                 8SVX
                ,
                 AIFF
                , or
                 WAV
                 files.  If the source file
    contains an annotation, it is copied; otherwise,
    oplay inserts its own.  This option is valid for
    ONE FILE ONLY.  To write multiple conversions, you
    must use '-W' before each file to be converted.
    Default: none.
```

1.18 -i

```
Option description:
  -i: Displays extra info on the current sound.  Best used
    in conjunction with "-s1".  Shows the sampling rate,
    buffer size, sound length, volume, and style (i.e.,
    signed/unsigned/u-law, 8-bit/16-bit) of the sound.
    Default: 0.
```

1.19 -_o

```
                Option description:
  -O: Say you want a nasty Workbench window to tell what's
    up in OmniPlay.  You can do this with output
    redirection.  However, the new
                 OMNIPLAY
                 environment
    variable, which can be used to store your very own
    OmniPlay defaults, can't use such a redirection since
    redirection is a shell feature.  This option is
```

```
    equivalent to file redirection, except that it can
    be parsed by OmniPlay instead of the shell.  One of
    the sample environment variables has an example of
    window redirection.  Default: none.
```

1.20 --

```
                Option description:
  --: If you've a sound file called "-whack", oplay would
    give you an error for 'oplay -whack' since
    1) 'hack' is not a number (see '
                 -w
                '), and 2) there's
    no filename to play.  Use "oplay --whack" to remedy
    this.  Basically, it means that the next argument is
    a file, not an option.  Default: none.
```

1.21 -_r

```
                Option description:
  -R: Some people like to play sounds over and over [and
    over [and over [...] ] ].  This is for them.  It
    behaves like the
                 -r
                 and
                 -i
                 (and other) options in
    that it remains in effect until switched off with
    '-R1'.  '-R0' tells oplay not to play stuff, and is
    only useful if you want to get info about a file, but
    not play it.
```

1.22 examples

```
                Examples:

> oplay
                 -r15000
                 IllBeBack
  play file "IllBeBack" at 15,000 Hz.

> oplay
                 -r12000
                 IllBeBack
                 -r 18000
                 Mr.Ed
  play "IllBeBack" at 12,000 Hz, then play "Mr.Ed" at 18,000 Hz.

> oplay
                 --bigboom
```

```
                -p75

                -c r
                sounds:loudcrash
  play "-bigboom" from the first available channel, wait 1.5
  seconds, and play "sounds:loudcrash" from the right channel.

> oplay
                -b10000
                df0:toilet.aiff
  set buffer size to 10,000 bytes and play "df0:toilet.aiff".
  Note: "df0:toilet.aiff" will play correctly even if it's
  not an AIFF file because of auto-detect and because you
  didn't specify a forced file type.

> oplay
                -v 16
                dialtone
                -v0
                busysignal
  play "dialtone" at 1/4 maximum volume, then play "busysignal"
  at whataver its prerecorded volume is.  If "busysignal" has
  no prerecorded volume, play it at full.

> oplay
                -p 1
                reallybig.snd
                -p-1
                normal.snd
  play "reallybig.snd" at priority 1, then "normal.snd" at
  priority -1.

> oplay
                -o

                -b1000
                evenbigger.wav
                -o0
                tiny.voc
  play "evenbigger.wav" in one shot.  "-b" option defines the
  new default buffer size, but this default is ignored until
  "tiny.voc" is played.  Buffers for "evenbigger.wav" are both
  half the size of the file plus one byte.
```

1.23 misc

```
               Miscellany:


                Residentability

                OMNIPLAY
                Environment variable
```

1.24 resident

Residentability:

OmniPlay is residentable. If you use it a lot, you can make it resident with the AmigaDOS "resident" command, as in:

```
> resident oplay
```

Residenting OmniPlay swallows about 24.5 K, so it's pretty much worth it if you intend to use it often.

1.25 omniplay

Environment variable:

The environment variable OMNIPLAY is now read prior to any command-line arguments. Use this to set global defaults for

```
  all invocations of oplay.  The environment takes the same
  format as the command line.  Maximum environment length is
  256 bytes.  The directory env/ contains examples of some
  OMNIPLAY variables you might want to use.
```

1.26 double-buffering

```
                Double-buffering:

  OmniPlay uses double-buffering for seamless playback.  This
  means that one buffer is played while the next one is filled
  from disk, then the second is played while the first is
  refilled.  The slower your input device, the larger your
  buffers should be.

  Technical note: the buffer-loading, including conversion from
  16-bit,
                 u-law
                ,
                 unsigned
                , or
                 little-endian
                , seems
  to be fast enough even at priority of -1.  If you find that
  this is not true on your machine, try bumping the priority a
  little.  If it's still not acceptable, you'll have to rely on
  oneshot ('
                 -o
                ') loading.  At present, the reloader
  is a function, so it carries the time overhead associated
  with calling a function.  I could move this inline, at a
  small increase in code size, to speed it up (though obviously
  I'd rather not if possible).


Process Priority:

  I have made OmniPlay pause by loading a large Imploded
  program during playback, but this is an expected side-effect
  which shouldn't occur often.  Raising the
                 priority
                 will cure
  this, too.  If you ever want to play a really long file while
  you might be doing some CPU-intensive activity, and you'd
  prefer good sound over fast math, you can place
                 -p
                 options
  around the filename -- for example,

  > oplay ... -p1 bigsound.wav16 -p0 ...

  (16-bit
                 WAV
                 files are possibly the slowest in
  conversion, since samples must be downshifted to 8 bits after
```

converting from
little-endian
words.)

1.27 autotyping

Autotyping:

Automatic type identification is performed by examining the sound file, so you almost never need to tell OmniPlay what format a sound is. Filename extensions are meaningless to OmniPlay. Autotyping of formatted sounds is fast, but autotyping unheadered sounds can take a while on big files. See the
-f
option.

1.28 powerpacker

PowerPacker support:

PowerPacker is a data compression and encryption format devised by Nico Francois several years ago. The idea was originally just to compress files on disk, but later on Nico developed the powerpacker.library so that files could be decompressed/decrypted by the software that attempts to read the file.

OmniPlay supports playback of PowerPacked sound files through this library, but only when using the automatic type-identifier (see the
Autotyping
section). You'll
need enough free storage in your t: directory to hold the decompressed file until it's through playing. You also need enough free RAM to perform the decompression. To decrunch PP sound files, you need to have powerpacker.library either in your libs: directory, or already loaded into the resident shared library list. If powerpacker.library is not available, OmniPlay will still work -- it just won't be able to play PowerPacked files (you'll get a notice if you try to play one).

1.29 channel

Channel support:

The Amiga sports four audio channels split into full-polar stereo. You can tell OmniPlay to play from either a "left" channel, a "right" channel, or "either" channel using the

```
                -c
                option.  Choosing a channel of "either"
causes OmniPlay to use the first channel it can get.
Pecking order is: ch. 0 (L), ch. 1 (R), ch. 3 (L), ch. 2 (R).
```

1.30 disableoveride

```
                Disabling command-line overrides:

Many of OmniPlay's option switches will affect all sound
files named after the option.  There may be instances in
which you'd like to disable an option you used previously in
a command, however, such as when the first sound to be played
needs to be amplified but the second does not.  Following is
a list of switches to be used to force OmniPlay to return to
its default state for the characteristic in question:

If you used:    Disable with:
------------    -------------

                -v
                      -v0

                -r
                      -r0

                -c
                      -ce

                -b
                      -b4000

                -o
                      -o0

                -p
                      -p-1

                -w
                      -w0

                -f
                      -fa

                -s
                      -s1

                -g
                      -g0

                -m
                      -m0

                -a
```

```
                      -a1

               -i
                      -i1

               -R
                      -R1

The
               -W
               option affects only the one file following it, and
has no default.

The
               -d
               option's default state is queried from your OS.  It
has no default as such.

The
               -O
               option's default is standard output, but there is no
way to return output to stdout once this option has been
used.
```

1.31 confusion

```
              .au and .snd confusion:

Files which end in .au can be one of several things:
1) headered u-law;
2) headered linear;
3) unheadered u-law.
If the "auto" type (i.e., default as in "oplay smile.au")
doesn't sound right, try "oplay -fl smile.au".  OmniPlay
should be able to get it right as long as there's a header.
Even if there's not, the automatic type-identifier should
discover that it's a u-law file and play as such.

Likewise, files which end in '.snd' are ambiguous.  A .snd
could be:
1) unsigned raw;
2) headered u-law;
3) headered 8-bit PCM;
4) headered 16-bit PCM;
5) signed raw.
Again, OmniPlay will probably get it right, but if it sounds
like garbage, try to force it with the
               -f
               option (use
whatever argument works).
```

1.32 weirdness

```
               U-law weirdness:

It is recommended that when playing
               u-law
               files (typically
ending in '.au' or '.snd'), you use the default of '-g0'.
Scanning the file, as explained in the description of the
'
               -g
              ' option, can take much time; my A3000 (68030,
data & cache burst on) took about 33 secs to scan a 1.2 M
u-law file.  Since scanning involves part of the u-law
decoding, the limiting factor on fast input devices such as
hard disks or RAM: is computation speed rather than access
time.  Obviously, then, it would take a slower Amiga more
time to scan.  Using '-g0' avoids the scan, enabling OmniPlay
to begin playback immediately.  As suggested before, there
can be some loss of volume and quality in this method; if
you're concerned about this, use '-g1' to get the best
possible quality.  You can use '-m1' to find out whether
you're really gaining anything by doing this; if the reported
maximum is 32124, the '-g0' default will produce the same
output as "-g1".  Maxima are logarithmic and increasing, so
if you're (logarithmically speaking) close to 32124, you're
getting nearly the best playback possible.

OmniPlay uses a precomputed logarithm-to-linear conversion
table for part of the u-law decoding process (not affecting
the final output at all), so u-law lag-time before playback
is about 55% less than the lag-time before output in most
u-law converters.  This table adds aproximately 1K to the
size of the OmniPlay executable.
```

1.33 intuition

```
               Intuition interface:

OmniPlay has no Intuition front-end, since I consider them
extraneous on such projects as this.  Again, let me know if
this is essential to you.

You can simulate an Intuition window display by redirecting
standard output to a CON: window, e.g.,

> oplay K00L.biff <>CON:0/0/640/60/OmniPlay

See also the
               -O
               option.
```

1.34 stopping

Stopping OmniPlay's playback:

To break OmniPlay during playback, send it a C-break (SIGBREAKF_CTRL_C) either by hitting CTRL-C in the CLI input stream it started from, or by issuing the command "break <clinumber>", or with some system monitor such as ARTM. If oplay is playing a series of sound files, a single break will break the current sound, and an immediate second break will break the program.

1.35 palntsc

PAL vs. NTSC:

Because of the difference between PAL and NTSC electrical standards (i.e., PAL cycles at 50 Hz, and NTSC at 60 Hz), there is a sometimes-audible difference in playabck between a machine running on PAL current and one using NTSC power. OmniPlay is sensitive to the difference between PAL and NTSC in sound playback. It determines what system you have by querying GfxBase, who is usually right. However, GfxBase only knows what current the video display is using. With the newer Amiga custom chips, it's possible to make the video use a standard that you're not actually getting out of the wall socket. The result is that GfxBase can tell you that your NTSC Amiga is in PAL if that's how you set your chips, but the power in the audio chips is still NTSC. In such a case, you need the '
-d
' option so that you can force
OmniPlay not to believe everything it hears from GfxBase.

1.36 recompiling

Recompiling OmniPlay:

OmniPlay was written in C. It was originally written under DICE, but several versions ago I switched to SAS/C. At first I verified that it would compile under either package, but this is no longer a guarantee. It compiles very nicely with SAS, but you're on your own with anything else. I suspect that problems with DICE would be minimal, however, as long as you remove the pragmas and link with amigas.lib.

OmniPlay should be easily portable to other Amiga C environments; if you perform the changes to make it compile under another system, please send me your version and I'll try to make the distribution work for them all.

Anyone is free to recompile OmniPlay, subject to the conditions stated in the

Legal Stuff
section of this
manual.

1.37 legal

Legal Stuff:

(1) The unmodified program designated "OmniPlay", or "oplay" for short, as well as any product of the union between any compiler and the unmodified source code), is freely distributable. If you like it and have a need for it, then you're morally obligated to use it. You may spread it anywhere, any time, any way you choose. You can even make changes to the binary and distribute them separately (see (2)), but I'd prefer that you don't.

(2) Modified copies of the program, as produced by binary patching or by disassembly, recoding, and reassembly, may be distributed freely, but only if the name of the resulting binary file is not labelled "OmniPlay" or "oplay", or contains some word or phrase such as "patch" to indicate that it is not the original OmniPlay.

(3) Source code, except where noted as the product of another mind, is copyright ©1993 by D. Champion. It may be freely copied, distributed, modified, disemboweled, shredded, or reconstructed for any purpose, public or private, commercial or free, excepting those prohibited in (2).

(4) Source code may be included as part of other works only if the following notice is included in printed or electronic documentation:
  Portions of this program are copyright ©1993
  by D. Champion; All rights reserved.
and if relevant sections of the including code bear a similar notice. The symbol '(c)' may be substituted for the '©' symbol when necessary.

Basically, the program is yours, but the source code and names are mine. You have permission to copy components of the source code for any use, but only if the copyright statement is propagated.

This statement supercedes previous declarations of the code as Public Domain; technically, I suppose that previous incarnations of the program are still PD, but this and future versions will reserve copyright.

If you doubt whether a particular usage is fair, contact me anyway. My primary concern in placing restrictions on duplication is avoiding unnecessary confusion to the user and ensuring the availability of the code to everyone.

OmniPlay is presented with no guarantee of functionality or

safe operation. In using this program, you agree that I am absolved of any and all liability. This said, I have made efforts to protect OmniPlay's integrity by testing under Enforcer and Mungwall, and I'm pretty sure nothing rotten is going to happen. If something rotten does happen, please let me know. I'll do my best to provide prompt bug fixes, and to update OmniPlay's format support as I receive new format information. If you have a format you'd like to see supported, tell me about it. Send me some samples in that format. Point me to the specs. I'll work on it. (I've heard that one floppy can be sent in a standard envelope through U.S. Mail first-class postage.)

1.38 bugs

Known bugs in version 1.231:

- I've made OmniPlay agree with every VOC, WAV, or AIFF file that I own, but it's possible that it doesn't work with them all. In particular, compression and VOC silences are not supported since I have no documentation on how these are decoded. Please send me any information you have on these.
- Stereo playback is still busted. As the command line goes, asking for "stereo" equates to asking for "either". I think I know how to fix this now, but I don't know when I'll have the time to do it.

1.39 credits

Credits:

Thanks for code and suggestions to:

- Guido van Rossum, for the raw audio type-guesser, and for the
  Usenet
  comp.binaries.sounds.d Audio Formats Guide.

- Dave Schreiber, for DSound source code's availability. This is where I started learning to program the audio.device, and my playroutine is largely modelled after his.

- Nico François, for the powerpacker.library and PowerPacker compression format.

- Malcolm Slaney and Ken Turkowski of Apple Computer for ConvertIeeeExtended(), the basis for the AIFF rate converter in OmniPlay, which AudioMaster desperately needs.

- Michael Boehnisch, author of ulawto8svx, for the

```
    reproduction of the u-law decoder;

  - Brian Foley, for the u-law decoder itself.

  - All contributors to SOX, the Sound Exchange, for the idea
    of a multi-format player and for example code.

  - Chris Wichura, Amish Dave, and Matt Simmons for pre-release
    comments and suggestions.  Special thanks to Chris for his
    many suggestions on how to implement stuff I wasn't
    familiar with, and for good example source code.

  - Dan Barrett for several reports on nasty bugs, and for
    providing the information to help me track them.

  - Authors of quality free software, for quality free
    software.
```

1.40 author

```
Contacting the author:

  You can reach me at:

  David Champion
  951 E. 54th Place #3
  Chicago, IL 60637
  U.S.A.

  (Internet, for a long time yet)
  dgc3@midway.uchicago.edu

  Previously disclosed e-mail address might not continue to
  exist, so don't use them.
```

1.41 changes

```
                Changes since last version:

1.23 -> 1.231

  Released: 31 Jan 93


                 v1.23
                 failed to deallocate one buffer plus 72 bytes when both
  buffers were used.  This is fixed; all allocated memory is
  now freed.

  ,
                 -wn
                ' option timed out for n seconds, not n/50 seconds as was
```

documented. This is fixed; '-w' now counts in fiftieths of a
second.

,
-R
' option added. Now you can play something a lot of times.
This is something of a kludge, though -- the files must still
be reloaded each time. This means that you can't delete a
file that's being replayed, or oplay will have a cardiac
arrest. (The OS will lock the file from deletion while it's
open, so you should have no problems. Just don't use any
system-zappers like ARTM or XOper to unlock the file, okay?)

Documentation reformatted (and partially rewritten) as an
AmigaGuide
database. This is a plus, whether you believe it
or not.

1.22 -> 1.23

1.21 -> 1.22

1.20 -> 1.21

1.15 -> 1.20

1.10 -> 1.15

1.01b -> 1.10

1.01 -> 1.01b

1.00 -> 1.01

0.98 -> 1.00

0.975 -> 0.98

0.97 -> 0.975

0.96 -> 0.97

0.?? -> 0.96

0.??

1.42 1.22 -> 1.23

1.22 -> 1.23

Released: 19 Nov 92

Egad. More ugly stuff. Fixed two bugs that made oplay's

```
style-identifer sometimes freeze the task with a division by
zero. (Yeah, I admit it was stupid.)  The autotyper could
freeze under two conditions: first, if the amplitude range
was under half the resolution, and second, if a raw-audio
file appeared to be
               u-law
              .


               Priority
               handling bad.  OmniPlay would not return its CLI to
the original priority.

Changed the '
               --
              ' option syntax.  '---file' now tries to play
'--file', rather than '-file'.  Also fixed a bug that made
oplay ignore any arguments after a '--' option's argument.

Changed operation of '
               -o
              ' (one-shot) option.  Previously set
buffers to half of the sample length for memory efficiency,
but this caused delays when playing small files from slow
devices or when playback required much math.  Now sets
buffers to exactly the sample length, but doesn't allocate
the second buffer at all.

Added version information to default output.

Recompiled using pragmas instead of amiga.lib stubs.  This
makes a smaller, faster program.
```

1.43 1.21 -> 1.22

```
              1.21 -> 1.22

Released: 12 Nov 92

(This was released as a patch archive containing the files
'oplay', 'src/oplay.c', 'CHANGES.doc', and 'pch121-122.diff',
a context-diff to convert the v1.21 source to v1.22.)

Fixed a bug wherein OmniPlay caused Enforcer hits (four long
reads from address 0) if no
               OMNIPLAY
               environment variable was
present.  The hits did not occur when OMNIPLAY existed.  This
was painful enough to Enforcer users to warrant a small fix.
```

1.44 1.20 -> 1.21

```
              1.20 -> 1.21

Released: 29 Oct 92

Well, I thought OmniPlay was
              Kickstart 1.3
              -compatible, but I
was wrong.  I forgot about the ExamineFH() that gets called
for
              .voc
               and raw files, and about the FGetC I used in a
couple of other places.  They're gone now.  1.3 users rejoice
(but not for long).

Added support for an
               OMNIPLAY
               environment variable.

Added
               -O
               option to specify output file.  This is for use in
the OMNIPLAY environment variable.

Fixed
               PowerPacker
               bug.  Now plays any number of Power-Packed
files, in any order.  (See BUGS in the documentation to any
older version.)


               -a
               did not operate in accordance with the documentation.
Now it does.
```

1.45 1.15 -> 1.20

```
              1.15 -> 1.20

Date: 20 Oct 92 -- Not released

Added
               -W
               option.  Amish thinks that run-time sound
conversion is a waste if you already know what it sounds
like. Maybe so for slow CPU's, but I still prefer not
converting the disk file.
```

1.46 1.10 -> 1.15

```
              1.10 -> 1.15
```

```
Date: 18 Oct 92 -- Not released

Added
                -i
                option because I wanted more info on my sounds.
```

1.47 1.01b -> 1.10

```
               1.01b -> 1.10

Released: 26 Sep 92

Added
                -a
                option so quiet sounds can be made louder.
```

1.48 1.01 -> 1.01b

```
               1.01 -> 1.01b

Date: 25 Sep 92 -- Not released

Stereo playback (
                -cs
               ) was disabled in 1.01 because it
didn't work correctly -- it seems to always play from the
left channel. It's been pointed out, however, that removing
this option could cause people to have to change existing
scripts, while leaving it in does no harm.  It's back now.  I
removed the documentation of this option argument in 1.01,
and haven't reinstated it along with the arg itself because,
at this point, it's strictly for backward-compatibility.
```

1.49 1.00 -> 1.01

```
               1.00 -> 1.01

Released: 23 Sep 92

Fixed a bug in the playroutine that made using buffers lots
bigger than the sound file sort of forget the last bufferful,
thereby chopping off the last bit of sound.

Added
                -o
                option so sounds can be fully loaded before
playback begins.  This is only useful on slow input devices
like floppies and slow NFS-mounted volumes.

Modified the help text.
```

1.50 0.98 -> 1.00

```
0.98 -> 1.00

  Not released. Not really existent, to be honest.  I just
  wanted to name the real next release to be 1.01 so you'd know
  it's a bug-fix.
```

1.51 0.975 -> 0.98

```
0.975 -> 0.98

  Released: 3 Sep 92
  First release.  Nothing new.
```

1.52 0.97 -> 0.975

```
               0.97 -> 0.975

  Fixed the false optimization to u-law playback in v0.97, and
  substituted the precomputed log table.  Added
                -g
                option to
  tell oplay whether to use it, and
                -m
                to help you decide.
```

1.53 0.96 -> 0.97

```
               0.96 -> 0.97

  Date: 29 Aug 92

  Optimized the
                u-law
                player stuff so that scans of the u-law
  range were faster.  As it turns out, this "optimization" was
  bogus, and caused oplay not to decode u-law at all.
```

1.54 0.?? -> 0.96

```
               0.?? -> 0.96

  Things get really hazy past this point.  I added the

                PowerPacker
                decompression somewhere in here.
```

1.55 0.??

```
0.??

  Date: Late August 1992

  All the basics.
```

1.56 todo

```
                Things to be done in future versions:

  - Still working on stereo stuff.  It's not that it's so hard,
    but that it's generally not important, and I want to get
    this thing 2.0-ized.
  - Fix minor problem in new
                -R
                option.
  - Some day I might port the file reading mechanism to a set
    of KS-3.0 datatypes, but this would require that I be told
    how to write a datatype.  Hint, hint.
  - Implement 14-bit sound through volume modulation of one
    channel by another.

Note Well:

  (so I lied last time, and the time before, and ...)
  Very soon, I will begin rewriting (parts of?) OmniPlay such
  that it WILL NOT WORK under any OS version less than 2.04.
  New improvements will rely on v37+ OS calls:
  - Future use of ReadArgs() to parse options.
  - v37+ w/ ReadArgs() handling of Workbench Tool Types.
  - A special treat I'm not discussing now.
  - Some other things I don't remember right now.
  - ARexx port so oplay can be a daemon.  (Not strictly 2.0, but most
    ARexx users got it with 2.0.)

  On the other hand, I might release an oplay that does stereo
  before a 2.0-only version.  It depends on which gets done
  first, and how far behind the other is when the first is
  done.
```

1.57 amigaguide

```
AmigaGuide:

  A nice hypertext utility from Commodore.  It uses data files
  like this one here.  I just wish they'd release it to
  everyone so no one had to muck around with ShowHyp and Hyper
  and cheesy things like that.
```

1.58 ks13

Kickstart 1.3:

A really old operating system. Yuck. OmniPlay still works under it, though. Well, it does in theory. I never run 1.3, so I'm not sure.

1.59 formats

OmniPlay supports the following standards:

IFF-8SVX

IFF-AIFF

Creative Voice File (.VOC)

RIFF-WAVE (.WAV)

Sun audio (.au)

NeXT audio (.snd)

Raw (Unformatted)

Signed Bytes

Unsigned Bytes

U-Law (14-bit encoded to 8 bits)

1.60 8svx

IFF-8SVX:

Signed (two's complement) 8-bit PCM samples with IFF header. Allows for storage of Rate, Volume, and Channel information. Also supports several optional properties such as Name, Author, Copyright, and Annotation.

Usually found on Commodore Amiga microcomputers.

1.61 aiff

IFF-AIFF:

Signed (two's complement) PCM samples with IFF header. Allows rate precision to many decimal points. Supports both 8-bit and 16-bit audio. Supports IFF Name, Author, Copyright, and Annotation properties.

Usually found on Apple Macintosh and IIgs, and on SGI Iris and Indigo.

1.62 voc

Creative Voice File:

Unsigned 8-bit PCM samples. Integral sample rates that are factors of 1,000,000. Supports silence compression, but OmniPlay doesn't.

Usually found on Intel-based micros sporting a SoundBlaster audio card. Blech.

1.63 wav

RIFF-WAVE:

Unsigned n-bit PCM samples. Any number of bits allowed; usually 8. If more than 8 bits, data is in Intel
(
little-endian
) byte order. Arbitrary integral rates, and many optional properties. Some properties are transferred to IFF-like properties by OmniPlay's display and
8SVX
converter.

Usually found on Intel-based micros (PC Clones) running Microsoft Windows 3.1. Icky.

1.64 au

Sun audio:

Usually 14-bit PCM encoded to 8-bit u-law. Usually 8000 Hz sampling rate. I know of no exceptions, but the Sparc 10 has 16-bit audio and I don't know how it's stored. Might be raw audio, but might have Sun header. The Sun header is identical to the NeXT header except that it allows ONLY u-law and ONLY 8000 Hz.

Usually found on Sun Microsystems SparcStations. Yay.

1.65 snd

```
                NeXT audio:

  An extension of Sun's audio header, allowing many sample
  styles and arbitrary rate.  The most common NeXT files are
  8-bit u-law, 8-bit PCM, and 16-bit PCM; NeXT u-law is
  sampled at 8012, not 8000, Hz.

  Found on NeXT workstations.

  The '.snd' extension is widely used on many platforms for
  many types of sound.  Do not assume that a file called
  'whop.snd' is a NeXT sound.  OmniPlay will usually figure it
  out for you, but in case you need to know, any '.snd' file
  can be:
  - NeXT something
  -
                 Signed
                 8-bit Raw
                 PCM
                 (Amiga, SGI)
  -
                 Unsigned
                 8-bit Raw PCM (Mac, Atari, IBM)
  - Raw
                 U-Law
                 (Sun)
```

1.66 ulaw

```
U-Law Encoding:

  A method of compressing 14-bit sound to 8 bits.  Used almost
  exclusively on Suns, NeXTs, and some other Unix machines.

  Future versions of OmniPlay may support playback in full
  14-bit sound by using one audio channel's volume control to
  modulate another channel.  This would allow 16-bit sounds
  to be played in 14 bits rather than merely 8, as well.
```

1.67 pcm

```
Pulse Code Modulation (PCM):

  Basically, PCM samples are raw numbers generated by an
  analog-to-digital converter (ADC).  They can be fed directly
  into a digital-to-analog conveter (DAC) to reproduce the
  original sound.  PCM is a linear scale, so that a sample
  value of n is half as loud as one of 2n.

  Most systems store their samples in some PCM format; these
```

are usually either signed or unsigned, depending on how the computer's audio mechanism works.

Signed PCM defines a zero value as silence, with samples ranging from -128 to +127 (in 8-bit audio).

Unsigned PCM defines 128 as silence, with a range of 0 to 255 (in 8-bit).

1.68 little-endian

Little-Endian:

This term desribes the byte order of values larger than 255 in Intel microprocessors. It means that the smaller byte -- the byte representing 2^0 through 2^7 -- comes first, followed by the larger byte -- the one representing 2^8 through 2^15. If four bytes are needed to describe a value -- that is, if the value is greater than a 16-bit number -- then the lowest byte is first, then the next-to-lowest, then the next-to-highest, then the highest.

Audio files originating on a computer using an Intel processor will be in little-endian format, and will take some time to convert to the Motorola ("big-endian") format used by the Amiga, Macintosh, Atari, NeXT, and SGI. Other machines which do not use Motorola CPUs (e.g., Sun Sparcs) usually use the Motorola byte order.

1.69 usenet

Usenet:

A virtual association of computers which communicate regularly to advance the spread of information. Details are spared here, but one of the characteristics of Usenet is its news system -- a means of ordering public communications under a large number of "newsgroups" such that information on a given topic can be assigned to a more or less small number of areas of interest. It's like a really huge BBS with hundreds of thousands of users and several thousand message areas.

This is something you desire. Honest.

1.70 kbsp

Killer Buttkick Sound Player, version 2.1:

You're joking, right? Well, I'm joking.

1.71 mbsc

MegaMondo BigTime Sound Converter:

Slices, dices, thrashes, and trashes ANY SOUND into ANYTHING ELSE!!! 8SVX to WAVE. VOC to BIFF. AIRCRAFT TAKEOFF TO MOANS OF DELIGHT!! YOU NAME IT, IT MAIMS IT!! Want your phone conversation with the buyers to reveal insider trading? Just send MBSC into CRIMINAL ALLUSIONS MODE. Your mother-in-law's nagging to cries of approval? Use the special VOCAL INTENT WARP! There's no sound MBSC cannot turn into something COMPLETELY DIFFERENT!!! With a little effort, you can even turn a CONGRESSIONAL HEARING into a BBC RADIO COMEDY!!!! MBSC DOES IT ALL! What? No sampler? NO PROBLEM! Just jack MBSC's optional AURAL INTERFACE MODULE into your SEMICIRCULAR CANALS and pick up the vibrations!!! Even better is our upcoming NEURAL PICKUP DEVICE. It's like an electric blanket, but once you WRAP IT AROUND YOUR HEAD LIKE A TOWEL -- Wow, just watch out! Even your MOST PERSONAL THOUGHTS can be BROADCAST through MBSC's marvelous MULTIMEDIUM ADAPTER to send to ANY RADIO, TELEVISION, DISK FILE, or STADIUM in NANOSECONDS!!! With all this, what more could you want? How about a CONVERSATION with MBSC's ARGUMENTATIVE AI UNIT? How about SOMEONE ELSE'S conversation? It's IN THERE! MBSC is so powerful that it can even POUND THE AIR MOLECULES into DEKABELS of PURE AURAL HORROR!!! Even the longtime great AUDIOMONGER is no match -- we DESTROYED their CUSTOMER SERVICE DEPARTMENT with THEIR OWN INNERMOST LINGERING THOUGHTS!!!! Don't delay, CALL NOW!

1.72 sox

SOX: The Sound Exchange:

A very good sound converter. Converts most existing formats to most existing formats. Originally for Unix, it's been ported to AmigaDOS, VMS, and MS-DOS; with release 6, it compiles readily with no changes under any of these. If you really have to convert sounds, SOX is all you need. For those with Internet access, Amiga SOX can be retrieved from amiga.physik.unizh.ch (and AmiNet mirrors) in the directory mus/edit. Release 5 (Amiga version 2.0) is named "amisox20bin.lha", but release 6 will be available shortly.

1.73 spiff

Searing Power Intermedium File Format (SPIFF):

The primary format of the 41 custom formats supplied with the MegaMondo BigTime Sound Converter (

MBSC

). Unfortunately,
only MBSC itself has the raw power necessary to even find a
sample in this format. MBSC is really the only way to go,
but if you can't stand the unparalleled, uncompromised
ferocity of MBSC, then OmniPlay will do for most material.