Table of Contents

The Blog

Playing .mod tracker music

The sound of 16 bit. When programmers made music.

From the era when you had CPU power to just play PCM samples.

Not enough CPU to decompress a full file, but not enough storage to store a full PCM track.

Enter mod/xm3 files.

It's like midi but way more cooked.

You pick a set number of PCM samples which you will use to make your music.

Say a piano sound, a drum sound, so on. Encode those 8 bit or 16 bit PCM.

Then, you have a set number of channels, 4 for mod.

You pick which sample to load, at what volume, and what pitch. That's all.

So, 4 things play at once, you have 16 sounds you picked, you can make a full song!

You can use things like miltytracker but that is boring.

How small would the files and a decoder be? I wanted to try this on a low power microcontroller.

But first, the proof of concept in Linux.

In 500 lines, I was able to read the file and pipe it to ALSA.

Compiled binary on Linux is 21 KiB, meaning without alsa and on microcontroller, we can make it happen.

The song file I'm using is also only 60 KiB! I'm sure there may be a way to turn MIDI into MOD adding more music.
Or make a neural net to pick 16 samples out of a song and create the .mod file.

It's not perfect, it only plays a subset of files, and my timings are off, but not terrible for an hour of effort.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <math.h>
#include <alsa/asoundlib.h>
```

```
// MOD file constants
#define MAX_SAMPLES        31
#define NUM_CHANNELS       4
#define ROWS_PER_PATTERN   64
#define BYTES_PER_NOTE     4
#define SAMPLE_RATE        44100
#define BUFFER_SIZE        1024
#define BASE_TEMPO 125  // Default MOD tempo in BPM

// Note period table for Amiga frequency conversion
const uint16_t period_table[] = {
    // C    C#   D    D#   E    F    F#   G    G#   A    A#   B
    856, 808, 762, 720, 678, 640, 604, 570, 538, 508, 480, 453, // Octave 1
    428, 404, 381, 360, 339, 320, 302, 285, 269, 254, 240, 226, // Octave 2
    214, 202, 190, 180, 170, 160, 151, 143, 135, 127, 120, 113  // Octave 3
};

// Pattern data structure
typedef struct {
    uint8_t sample;    // Sample number (0-31)
    uint16_t period;   // Note period
    uint8_t effect;    // Effect number
    uint8_t param;     // Effect parameter
} Note;

// Sample data structure
typedef struct {
    char name[22];
    uint16_t length;           // Length in words (2 bytes)
    uint8_t finetune;          // Finetune value (0-15)
    uint8_t volume;            // Default volume (0-64)
    uint16_t repeat_point;     // Repeat point in words
    uint16_t repeat_length;    // Repeat length in words
    int8_t *data;              // Sample data (8-bit signed)
} Sample;

// MOD file structure
typedef struct {
    char title[21];
    Sample samples[MAX_SAMPLES];
    uint8_t song_length;       // Number of positions
    uint8_t positions[128];    // Pattern sequence
    uint8_t num_patterns;      // Number of patterns (calculated)
    Note (*patterns)[ROWS_PER_PATTERN][NUM_CHANNELS]; // Pattern data
} MODFile;

// Channel state
typedef struct {
```

```
    uint16_t period;           // Current note period
    uint8_t sample_num;        // Current sample number
    uint8_t volume;            // Current volume (0-64)
    uint32_t sample_pos;       // Position in sample (fixed point: 16.16)
    uint32_t sample_increment; // Sample position increment per tick
    uint8_t effect;            // Current effect
    uint8_t param;             // Effect parameter
    uint8_t porta_target;      // Portamento target period
    uint8_t vibrato_position;  // Vibrato wave position
    uint8_t tremolo_position;  // Tremolo wave position
} ChannelState;

// Read a 2-byte big-endian value
uint16_t read_big_endian_16(const uint8_t *data) {
    return (data[0] << 8) | data[1];
}

// Convert Amiga period to frequency
float period_to_freq(uint16_t period) {
    if (period == 0) return 0;
    return 7159090.5f / (period * 2);
}

// Print a message and exit if ALSA returns an error
void check_alsa_error(int err, const char *msg) {
    if (err < 0) {
        fprintf(stderr, "%s: %s\n", msg, snd_strerror(err));
        exit(EXIT_FAILURE);
    }
}

// Add this function before main()
int calculate_tick_samples(int tempo) {
    // Calculate samples per tick based on tempo
    // 2500 / tempo = milliseconds per tick
    // (ms * sample_rate) / 1000 = samples per tick
    return (2500 * SAMPLE_RATE) / (tempo * 1000);
}


// Load a MOD file
int load_mod_file(const char *filename, MODFile *mod) {
    FILE *file = fopen(filename, "rb");
    if (!file) return 0;

    // Get file size
    fseek(file, 0, SEEK_END);
    long file_size = ftell(file);
```

```
    fseek(file, 0, SEEK_SET);

    // Read the entire file into memory
    uint8_t *data = (uint8_t*)malloc(file_size);
    if (!data) {
        fclose(file);
        return 0;
    }

    fread(data, 1, file_size, file);
    fclose(file);

    // Read title
    memcpy(mod->title, data, 20);
    mod->title[20] = '\0';

    // Read sample information
    uint8_t *ptr = data + 20;
    for (int i = 0; i < MAX_SAMPLES; i++) {
        memcpy(mod->samples[i].name, ptr, 22);
        mod->samples[i].name[22] = '\0';
        ptr += 22;

        mod->samples[i].length = read_big_endian_16(ptr) * 2; // Convert to
bytes
        ptr += 2;

        mod->samples[i].finetune = *ptr++;
        mod->samples[i].volume = *ptr++;

        mod->samples[i].repeat_point = read_big_endian_16(ptr) * 2; // Convert
to bytes
        ptr += 2;

        mod->samples[i].repeat_length = read_big_endian_16(ptr) * 2; //
Convert to bytes
        ptr += 2;
    }

    // Read song information
    mod->song_length = *ptr++;
    ptr++; // Skip unused byte

    memcpy(mod->positions, ptr, 128);
    ptr += 128;

    // Skip 4 bytes (format identifier M.K. or similar)
    ptr += 4;
```

```
    // Determine number of patterns
    mod->num_patterns = 0;
    for (int i = 0; i < mod->song_length; i++) {
        if (mod->positions[i] > mod->num_patterns) {
            mod->num_patterns = mod->positions[i];
        }
    }
    mod->num_patterns++;

    // Allocate and read pattern data
    mod->patterns = (Note (*)[ROWS_PER_PATTERN][NUM_CHANNELS])malloc(
        mod->num_patterns * ROWS_PER_PATTERN * NUM_CHANNELS * sizeof(Note));

    for (int p = 0; p < mod->num_patterns; p++) {
        for (int r = 0; r < ROWS_PER_PATTERN; r++) {
            for (int c = 0; c < NUM_CHANNELS; c++) {
                uint8_t b0 = *ptr++;
                uint8_t b1 = *ptr++;
                uint8_t b2 = *ptr++;
                uint8_t b3 = *ptr++;

                // Decode note data
                uint16_t period = ((b0 & 0x0F) << 8) | b1;
                uint8_t sample = (b0 & 0xF0) | (b2 >> 4);

                mod->patterns[p][r][c].period = period;
                mod->patterns[p][r][c].sample = sample;
                mod->patterns[p][r][c].effect = b2 & 0x0F;
                mod->patterns[p][r][c].param = b3;
            }
        }
    }

    // Read sample data
    for (int i = 0; i < MAX_SAMPLES; i++) {
        if (mod->samples[i].length > 0) {
            mod->samples[i].data = (int8_t*)malloc(mod->samples[i].length);
            memcpy(mod->samples[i].data, ptr, mod->samples[i].length);
            ptr += mod->samples[i].length;
        } else {
            mod->samples[i].data = NULL;
        }
    }

    free(data);
    return 1;
}
```

```
// Release MOD file resources
void free_mod_file(MODFile *mod) {
    for (int i = 0; i < MAX_SAMPLES; i++) {
        if (mod->samples[i].data) {
            free(mod->samples[i].data);
            mod->samples[i].data = NULL;
        }
    }
    if (mod->patterns) {
        free(mod->patterns);
        mod->patterns = NULL;
    }
}

// Render simple ASCII visualization
void render_visualization(ChannelState *channels, MODFile *mod, int position,
int row) {
    printf("\033[H\033[J"); // Clear screen

    // Display module info
    printf("Module: %s\n", mod->title);
    printf("Position: %d/%d  Pattern: %d  Row: %d/64\n\n",
           position, mod->song_length, mod->positions[position], row);

    // Display channels
    printf("Channel | Sample  | Period | Volume | Effect\n");
    printf("--------|---------|--------|--------|--------\n");

    for (int c = 0; c < NUM_CHANNELS; c++) {
        const char *sample_name = "<none>";
        if (channels[c].sample_num > 0 && channels[c].sample_num <=
MAX_SAMPLES) {
            sample_name = mod->samples[channels[c].sample_num-1].name;
        }

        printf("   %d    | %-7d | %6d | %6d | %X%02X  %s\n",
               c+1, channels[c].sample_num, channels[c].period,
               channels[c].volume, channels[c].effect, channels[c].param,
               sample_name);
    }

    // Simple volume meters
    printf("\nVolume Meters:\n");
    for (int c = 0; c < NUM_CHANNELS; c++) {
        printf("[%d] ", c+1);

        // Only show if channel is active
```

```
        if (channels[c].period > 0 && channels[c].sample_num > 0) {
            int vol_bars = (channels[c].volume * 30) / 64;
            for (int i = 0; i < 30; i++) {
                printf("%c", i < vol_bars ? '#' : '.');
            }
        } else {
            printf("<inactive>");
        }
        printf("\n");
    }

    fflush(stdout);
}

// Process one tick of audio
void process_tick(MODFile *mod, ChannelState *channels, int16_t *buffer, int
buffer_size) {
    // Clear buffer
    memset(buffer, 0, buffer_size * sizeof(int16_t));

    // Mix channels
    for (int i = 0; i < buffer_size; i++) {
        int32_t mixed = 0;

        for (int c = 0; c < NUM_CHANNELS; c++) {
            if (channels[c].period > 0 && channels[c].sample_num > 0) {
                int sample_idx = channels[c].sample_num - 1;

                if (sample_idx >= 0 && sample_idx < MAX_SAMPLES &&
                    mod->samples[sample_idx].data &&
                    mod->samples[sample_idx].length > 0) {

                    // Get current sample position
                    uint32_t pos = channels[c].sample_pos >> 16;

                    if (pos < mod->samples[sample_idx].length) {
                        // Apply volume and mix
                        int sample_val = mod->samples[sample_idx].data[pos];
                        mixed += ((sample_val * channels[c].volume) / 64) * 4;
// Increase gain by 4x

                        // Update position
                        channels[c].sample_pos +=
channels[c].sample_increment;

                        // Handle looping
                        if (mod->samples[sample_idx].repeat_length > 2) {
                            uint32_t loop_end =
```

```
mod->samples[sample_idx].repeat_point +
mod->samples[sample_idx].repeat_length;

                            if ((channels[c].sample_pos >> 16) >= loop_end) {
                                channels[c].sample_pos =
mod->samples[sample_idx].repeat_point << 16;
                            }
                        } else if ((channels[c].sample_pos >> 16) >=
mod->samples[sample_idx].length) {
                            // Stop playback if no loop
                            channels[c].sample_pos = 0;
                            channels[c].period = 0;
                        }
                    }
                }
            }
        }

        // Clamp and store in buffer
        if (mixed > 32767) mixed = 32767;
        if (mixed < -32768) mixed = -32768;

        // Scale appropriately for 16-bit output
        buffer[i] = mixed;
    }
}

// Process a row of the pattern
void process_row(MODFile *mod, ChannelState *channels, int position, int row)
{
    int pattern = mod->positions[position];

    // Process each channel
    for (int c = 0; c < NUM_CHANNELS; c++) {
        Note note = mod->patterns[pattern][row][c];

        // Process sample change
        if (note.sample > 0) {
            channels[c].sample_num = note.sample;
            channels[c].volume = mod->samples[note.sample-1].volume;
        }

        // Process note
        if (note.period != 0) {
            if (note.effect != 0x3) { // Skip if portamento effect
                channels[c].period = note.period;
                channels[c].sample_pos = 0;
```

```
                // Calculate sample increment based on period
                float freq = period_to_freq(note.period);
                channels[c].sample_increment = (uint32_t)((freq * 65536.0f) /
SAMPLE_RATE);
            }
        }

        // Save effect and param
        channels[c].effect = note.effect;
        channels[c].param = note.param;

        // Process effects
        switch (note.effect) {
            case 0x0: // Arpeggio
                // Handled in tick processing
                break;

            case 0xA: // Volume slide
                // Handled in tick processing
                break;

            case 0xC: // Set volume
                if (note.param <= 64) {
                    channels[c].volume = note.param;
                }
                break;

            case 0xD: // Pattern break
                // Handled by main loop
                break;

            case 0xF: // Set speed
                // Handled by main loop
                break;
        }
    }
}

int main(int argc, char **argv) {
    if (argc < 2) {
        printf("Usage: %s <input.mod>\n", argv[0]);
        return 1;
    }

    // ALSA setup
    snd_pcm_t *pcm_handle;
    int err;
```

```
    // Open PCM device for playback
    err = snd_pcm_open(&pcm_handle, "default", SND_PCM_STREAM_PLAYBACK, 0);
    check_alsa_error(err, "Unable to open PCM device");

    // Set parameters
    snd_pcm_hw_params_t *hw_params;
    snd_pcm_hw_params_alloca(&hw_params);

    err = snd_pcm_hw_params_any(pcm_handle, hw_params);
    check_alsa_error(err, "Cannot initialize hardware parameter structure");

    err = snd_pcm_hw_params_set_access(pcm_handle, hw_params,
SND_PCM_ACCESS_RW_INTERLEAVED);
    check_alsa_error(err, "Cannot set access type");

    err = snd_pcm_hw_params_set_format(pcm_handle, hw_params,
SND_PCM_FORMAT_S16_LE);
    check_alsa_error(err, "Cannot set sample format");

    err = snd_pcm_hw_params_set_rate_near(pcm_handle, hw_params, &(unsigned
int){SAMPLE_RATE}, 0);
    check_alsa_error(err, "Cannot set sample rate");

    err = snd_pcm_hw_params_set_channels(pcm_handle, hw_params, 1);
    check_alsa_error(err, "Cannot set channel count");

    err = snd_pcm_hw_params_set_buffer_size(pcm_handle, hw_params, BUFFER_SIZE
* 4);
    check_alsa_error(err, "Cannot set buffer size");

    err = snd_pcm_hw_params(pcm_handle, hw_params);
    check_alsa_error(err, "Cannot set parameters");

    err = snd_pcm_prepare(pcm_handle);
    check_alsa_error(err, "Cannot prepare audio interface for use");

    // Load MOD file
    MODFile mod;
    if (!load_mod_file(argv[1], &mod)) {
        printf("Failed to load MOD file: %s\n", argv[1]);
        snd_pcm_close(pcm_handle);
        return 1;
    }

    printf("Playing: %s\n", mod.title);
    printf("Song length: %d positions\n", mod.song_length);
    printf("Samples: %d\n", MAX_SAMPLES);
```

```
    // Initialize channel state
    ChannelState channels[NUM_CHANNELS] = {0};

    // Player state
    int position = 0;      // Position in the song
    int row = 0;           // Row in the pattern
    int ticks_per_row = 5; // Default speed (ticks per row)
    int current_tick = 0;  // Current tick within the row
    int tempo = 125;       // Default tempo (BPM)

    // Audio buffer
    int16_t buffer[BUFFER_SIZE];

    // Main playback loop
    while (position < mod.song_length) {
        if (current_tick == 0) {
            // Process new row
            process_row(&mod, channels, position, row);

            // Show visualization
            render_visualization(channels, &mod, position, row);

            // Check for pattern break effects
            for (int c = 0; c < NUM_CHANNELS; c++) {
                if (channels[c].effect == 0xD) {
                    row = ROWS_PER_PATTERN - 1; // Force next row to trigger
position change
                    break;
                } else if (channels[c].effect == 0xF && channels[c].param <=
0x1F) {
                    // Set speed (ticks per row)
                    if (channels[c].param > 0) {
                        ticks_per_row = channels[c].param;
                    }
                } else if (channels[c].effect == 0xF && channels[c].param >=
0x20) {
                    // Set tempo (BPM)
                    tempo = channels[c].param;
                }
            }
        }

        // Process and play audio for this tick
        process_tick(&mod, channels, buffer, BUFFER_SIZE);

        // Write to ALSA
        err = snd_pcm_writei(pcm_handle, buffer, BUFFER_SIZE);
        if (err == -EPIPE) {
```

```
            // EPIPE means underrun
            snd_pcm_prepare(pcm_handle);
        } else if (err < 0) {
            printf("Error from writei: %s\n", snd_strerror(err));
        }

        // Update tick counter
        current_tick++;
        if (current_tick >= ticks_per_row) {
            current_tick = 0;
            row++;

            if (row >= ROWS_PER_PATTERN) {
                row = 0;
                position++;

                if (position >= mod.song_length) {
                    break;
                }
            }
        }
    }

    // Clean up
    snd_pcm_drain(pcm_handle);
    snd_pcm_close(pcm_handle);
    free_mod_file(&mod);

    printf("\nDone!\n");

    return 0;
}
```

Just build with

```
gcc -o mod_player modplayer.c -lm -lasound

./mod_player popcorn_remix.mod
```

Some of my preferred test tracks:

- popcorn_remix.mod
- necroscope.mpd
- ELYSIUM.MOD

2025-05-31 08:48 · Tony

Tony's Proxmox Reinstall and Migration Guide

Use case

There are many guides for doing “migrations” or “reinstalls” of proxmox online, sometimes with backups, sometimes with live copies etc.

My needs

I have an existing Proxmox 7 server running with a single boot SSD using LVM for PVE. I have additional SSDs for hosting my LXC and VM images, and hard drives that are bind-mounted into the LXCs where bulk storage is needed (eg: Immich and Nextcloud data directories).

While this has backups of the containers, (and the bulk storage is RAID Z1), I am a single SSD failure away from the system not booting, which I consider catastrophic when it is a 6 hour $1000 flight away (thank Air Canada for that one).

I would like to reinstall PVE on this system, using two SSDs with ZFS RAID 1 for PVE, as 1) ZFS fits in far more with my current setup handling snapshots seemlessly, and 2) it gives me redundant boot and EFI partitions should one of the SSD's outright fail.

What is different

- I have NO quorum, I use a single instance of PVE.
- I can NOT do a live migration, as I have a single host, and will be reusing the hardware.
- I do NOT want to install all my LXCs and VMs from a backup - they live on a separate SSD I will not be wiping, and I would like to reuse those pools as is.
- I do NOT want to backup and restore PVE, I am changing SSD configurations (1 drive LVM to 2 drive ZFS)
- My old setup had many levels of jank during a PVE 7 to 8 upgrade, avoid re-using unnecessary old configs

Again, what I want, is to reinstall PVE onto a new SSD with a new filesystem, reuse my existing VM files and pools already on the drives. No backup/restore of LXCs. Just a simple, take everything, plunk it into new PVE.

The solution

Turns out the solution is actually relatively simple.

Since I tried a few things first and re-did this process twice, let me put some thoughts:

- Proxmox reads the configs for your VMs and LXCs from /etc/pve/nodes.

- I setup a full Proxmox Backup Server as an LXC another PVE instance (different location - not clustered)

The easy way

1. Make backups of everything you care about.
    1. IF YOU WILL BE RE-USING SSD'S, MAKE SURE YOU DD A BACKUP OF THEM!
        1. This one saved me and allowed me to repeat this procedure twice.
2. Grab the important config files you will need from the old host
    1. /etc/pve/hosts
        1. Ignore the keys and stuff, it'll cause pain and misery
    2. /etc/wireguard/wg0.conf
        1. If you rely on wireguard and want to be lazy and re-use old keys (not best security practice)
    3. /etc/fstab
        1. If you have non-standard mounts (I had an older btrfs HDD not in an zpool or lvm)
    4. /etc/exports
        1. If you have custom NFS exports to go from PVE over to VMs (LXCs just bind mount)
    5. /etc/network/interfaces
        1. If you have custom networking or VLAN setups
    6. ~/.ssh/authorized_keys
    7. Any special configs you have.
3. Swap out the SSDs for the new ones, prepare proxmox installation media
4. Install Proxmox fresh on your two new SSDs!
    1. You should be able to boot into your new system
5. Bring your system roughly in line with your old setup. For me, this included
    1. Re-setup network interfaces and bridges the way they were
    2. Restore my fstab, wireguard, exports
        1. Wireguard needs apt install wireguard wireguard-tools
        2. systemctl enable wg-quick@wg0.service –now (start your tunnel at boot)
6. Re-import your storage devices (eg: my VM/LXC SSDs, data HDDs)
    1. Make sure they are all detected under disks!
    2. For my data HDD zpool, this was two steps
        1. zpool import tonydata
            1. causes it to show up in node → disks
        2. add it in the proxmox UI
            1. datacenter → storage → add new → zfs
    3. For my LVM SSDs, this was 1 step as they were already listed under disks
        1. add it in proxmox UI
            1. datacenter → storage → add new → lvm thin
    4. For my old BTRFS HDD, this was 3 steps
        1. Restore my old fstab (verifying UUID stayed the same)
        2. mount -a
        3. datacenter → storage → add new → btrfs
            1. Remember to choose the types of storage this is (backup and data, not disk images in my case)
    5. Export my NFS pools (this one I forgot to do until my VM was mad)

        1. apt install nfs-kernel-server
        2. restore /etc/exports
        3. exportfs -av
7. Note, if you miss the above, it's probably not that bad, you'll be warned if things fail to start, it's iterative.
8. Restore /etc/pve/nodes/pve/lxc and /etc/pve/nodes/pve/qemu-server
    1. You should now see your left bar populate back up with all your past lxc and VMs
    2. Try starting them, maybe they work, maybe they error, if so look back to above.

That was it. Didn't end up using any of my backups from the backup server (surprisingly). Only used losetup on my dd to grab config files for networks and such retroactively to remember names and addresses.

Traps

- Do NOT, try to blindly restore your /etc/pve
    - Doing so will leave your /etc/pve in a bad state with keys that are lingering
        - You will NOT be able to log in and will need to systemctl stop, clear /etc/pve and such
- Connection error 401: permission denied - invalid PVE ticket
    - See my above point
- Do not try to RSYNC into /etc/pve
    - It will fail. It is a FUSE mount and will refuse the temp files rsync makes.
    - Use scp or an rsync flag to go direct to output file.
- Do NOT try to restore your PVE host from Proxmox Backup Server
    - Restoring the backup will yield errors with file overwrites
    - Allowing overwrites will break your configs in two and fail on /etc/pve
    - NOTE: proxmox-backup-client doesn't even backup /etc/pve by default (it only does root) so it's no use if you forgot that anyway
- PROXMOX BACKUP SERVER IS A RED HERRING!!
    - In fact, if you don't need to back up and restore VMs, and you've taken a DD of your old SSD, you don't need it.
    - it is excellent software for offsite backing up items, just not my current use case
- If it's not obvious, if you are using drive letters, like /dev/sda, sdb, YOU NEED TO STOP.
    - Please just use UUIDs, or heck even labels. After you reinstall, things WILL be shifting!!

In fact, let me do a whole segment on this

Proxmox backup server?

There seems to be some confusion with how this works, as a backup server can be added in two ways.

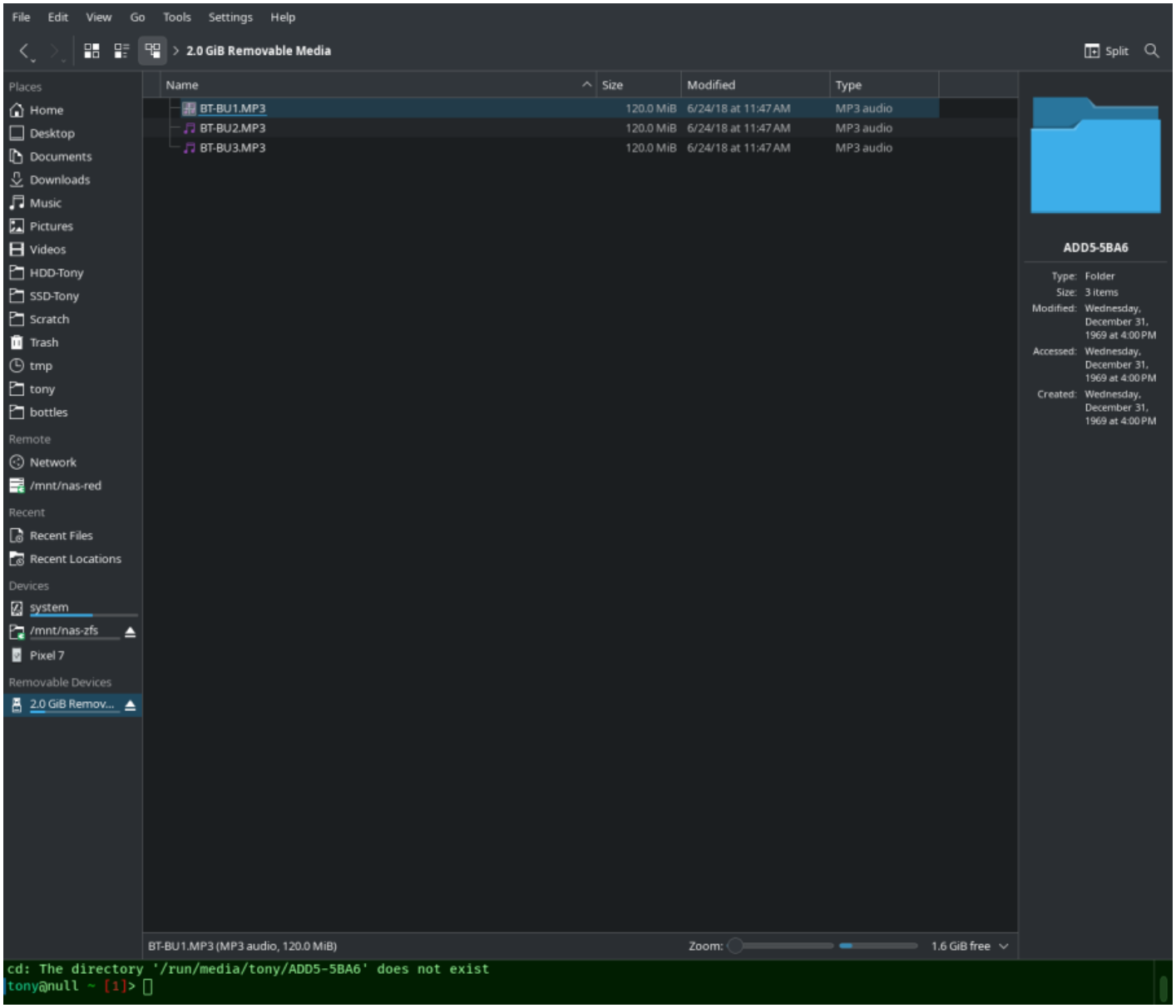
1. Datacenter → Storage
    1. This can be used for backing up the LXC and VMs on your system.
        1. Note: VM and LXC items backed up this way will keep some config data without /etc/pve, specifically ram and cpu amounts
    2. This can not be used to backup PVE itself

2. proxmox-backup-client
    1. This can be used to backup PVE. by default, backs up a partition of your choosing
    2. If you backup /, make sure you also include /etc/pve which is a fuse mount
        1. If you already DDed the drive, you can always losetup and mount the disk image instead of using pxar.

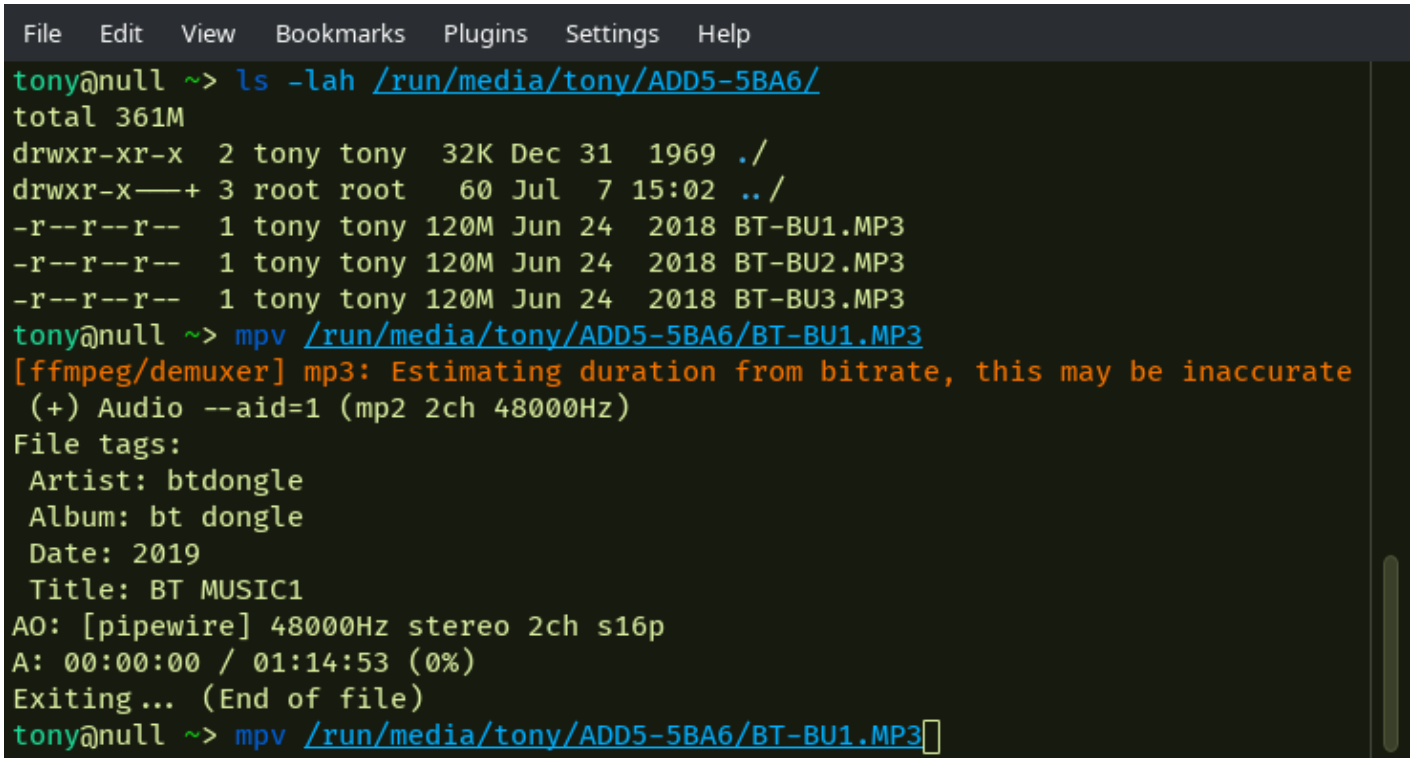
2025-01-03 05:22 · Tony

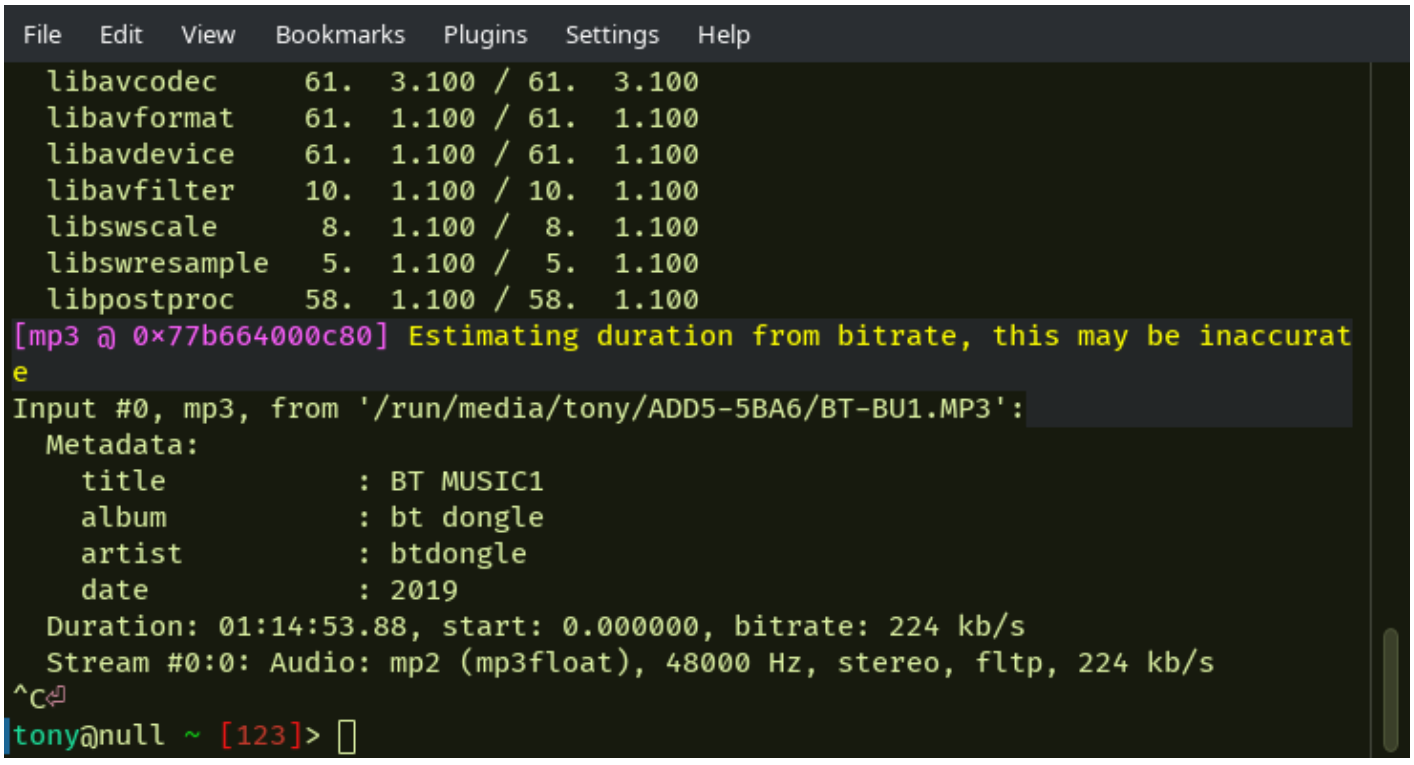
Weird USB Bluetooth Music receivers

My USB powered bluetooth audio receiver STARTED SHOWING UP AS A MASS STORAGE device.
I DIDN'T EVEN KNOW THAT THING HAD DATA LINES?!?
Did I get rubber-duckied by a BLUETOOTH RECEIVER?
WAIT YOU CAN MOUNT THE STORAGE DEVICE?!?



WTF IF YOU PLAY IT, THE FILE HAS LEGIT MP3 TAGS





OH SHIT I Think this is for it to work on crappy stereos that just play MP3 from a flash drive!!!
To adapt them to work with bluetooth
hm I don't hear anything and the mp3 file exits though
so I guess you need a dumber player that doesn't try to seek in a file
WAIT I THINK I KNOW WHAT THE OTHER FILES ARE
YOU CAN'T PLAY THEM, BUT IF YOU OPEN IT, IT TELLS THE PHONE TO GO TO THE NEXT SONG
so it's used to detect when you pressed previous/next!!! and from that, it signals to the bluetooth device

to go back/forward
who the f*** makes this ASIC?!? There's no way this thing is economies of scale

2024-07-07 22:18 · Tony

What is wrong with the 2023 League worlds opening ceremony?

The original league worlds opening ceremony video from this year has very many technical quirks. Here is what I noticed and spent a a few (prolly a dozen) hours fixing.

- The audio on the YouTube upload is unexplicably in mono and sounds notably worse than the Twitch livestream which was in stereo.
- Likely destructive interference when collapsing stereo to mono?
- The whole 'scripted' section of the opening is actually running at 29.97 fps, and interpolated to 60 but in the worst way.
- Half of their cameras are running at 29.97 which means every frame is duplicated for the stream which is 59.94 - this on it's own is fine. Most of the side-angle cameras are 29.97.
- The other half of the cameras are also running at 29.97, but are interpolated to 59.94 THROUGH FRAME BLENDING?!?
- This means that every other frame is literally a 50/50 blend of the two frames before it. This is what causes that motion-blur like feel, and you can instantly tell which cameras they are on.
- It might be the cameras that also do AR are doing blending? unsure.
- In fact, the main stage camera and the cable cam blend on opposite frames - eg: one has good even frames, blended off, the other is vice versa. This might be because of an internal delay? I'm not sure. Don't get me wrong 1080p29.97 is fine, although odd since sports is usually 720p60 or 1080i60.
- The version on YouTube took a video that had TV black levels, mapped it to PC, then was converted to TV again, resulting in crushed black and white levels losing detail
- The version on Twitch has the audio and video about 100 ms out of sync, this is most noticeable with the fireworks at the end of gods not being in line with music. Interestingly, the audio and video ARE synced correctly on YouTube version.
- TV (and most video formats) uses 16-235 to express black-white instead of 0-255, whereas PC is 0-255. So, if you take a 16-235 feed, map it to 0-255, then crop it into 16-235 again, you effectively increase contrast and reduce your dynamic range, that is what seems to have happened on YouTube.
- Another odd quirk: some of the camera angles are different between the YT and Twitch versions? Either these were cut after the fact, or it was filmed weird? Not sure, I can upload a side by side comparison if anyone is curious. Notable on some audience and a few side angle shots.
- Also if you watch the side by side, depending on if you sync up the video or audio feeds, it almost looks like there are 2 video systems, and depending on which you sync up on, some transitions happen 100 ms before or after the other, with the other half being synced. I think the AR stuff is probably separate?
- About that frame blending, there are 4k60 audience camera recordings on YouTube, and if you watch those carefully, you'll notice the camera feed going to the video walls also does frame blending. This tells me that whatever system video was going into first was doing the blended

frames, but again, only on half of the cameras.

So, to fix this

- I took the stream from Twitch, split the audio and video. For the video, I couldn't simply drop all odd or even frames because their different cameras interpolate different sets of frames.
- From my cut, the “main” stage camera had odd frames correct, and even frames blended, and the cable cam that is further away ad all even frames correct and odd frames blended.
- So I spent 3 hours clipping each camera angle, and dropping either the odd or even frames accordingly.
- I then re-synced the audio with that 100ms offset. This left me with a 29.97 fps video. That is my “cleaned up” version that I might upload, but, since the original was 60 fps, I wanted to interpolate it, but properly, not through frame blending.
- So I ran the whole thing through an AI video interpolation tool overnight on my GPU, and there you have it, 4k60 (technically 59.94). I'll add the 29.97 maybe as well.

Stuff I did not fix:

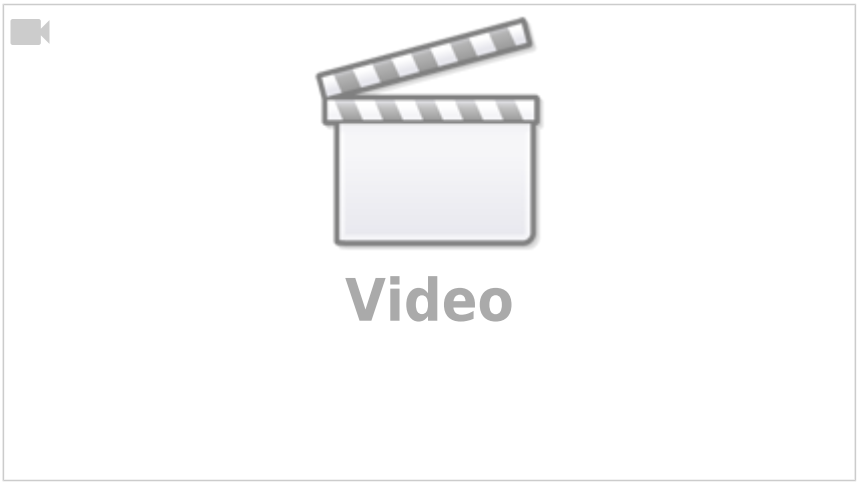
- The yellow jacket intro video is filmed in 24 fps, and brought up to 30, but not in a very consistent way, so it's staying as is.
- I can't change mic volumes, there isn't a lot, but you can pretty easily tell which parts are lip-synced anyway (listen for the reverb different, or when vocals sounds like they are stereo - eg: first 10s of vocals in gods sounds live, most of the rest is off the tape).
- I also can't fix their broken shadows on their AR characters.

Note: I do not own the music or video, those are from Riot Games. I'm simply uploading this in case anyone else wanted to watch it without the bizarre quirks of the version on YouTube.

If anyone from the Riot Games technical team is watching, I'm genuinely curious about how their production setup works and I would love to chat. (Or if you know someone who works with the technical team)

This is not meant to be a dunk on the performance, overall it was well put together, these are just 'quirks' I saw and couldn't unsee.

Here is the fixed version



2023-12-20 07:36 · Tony

Thinkpad T14 Gen2 AMD eDP Screen/Panel Upgrade

If you read Thinkpad T14 Gen2 AMD, I hated the screen they shipped with the T14, and thought it was a disgrace for a “premium” laptop.
Well, it annoyed me so much that I started to research how to upgrade the screen of the laptop.

Original Panel

The original panel is a was an Innolux N14HCA-EAE.
It is IPS-type, <250 nits, and has garbage colours.

Since it is 1080p non-touch, replacing it with another 1080p panel isn't too hard.
If you had a 4k screen, you would need a new eDP cable. If touch, replacing the digitizer is another step.

Look for the Thinkpad maintenance guide for removing an old screen. Follow it like Lego.
This is an irreversible process. Save the bezel. It's supposed to be replaced, good luck finding a replacement. Pry it slowly.
The original screen is held by double sided tape. I didn't replace mine, you maybe should - again, good luck.

New Panel

I bought the Innolux N140HCG-GQ2.
It's low power, 400 nit, 90+% sRGB, IPS-like, 1080. Much better.
No new cable needed, drop in replacement basically. Comparison sites label it the best of the 3/4 “low power” display options.
You can find via AliExpress and hope it's not a fake.

Re-use existing tape and bezel when reassembling, clip display properly.

Worth it?

100%. The brightness and colour gamut are very much noticable and appreciated.
It makes me actually enjoy looking at the laptop screen.

If you have the patience and courage, go for it! LMK how it went!

2023-07-02 15:42 · Tony

Older entries >>