typedef struct **MPContext** {

int osd\_show\_percentage;

int osd\_function;

const ao\_functions\_t \***audio\_out**;

play\_tree\_t \*playtree;

play\_tree\_iter\_t \*playtree\_iter;

int **eof**;

int play\_tree\_step;

int loop\_times;

**stream\_t \*stream**;

**demuxer\_t \*demuxer**;

**sh\_audio\_t \*sh\_audio**;

sh\_video\_t \*sh\_video;

**demux\_stream\_t \*d\_audio**;

demux\_stream\_t \*d\_video;

demux\_stream\_t \*d\_sub;

mixer\_t mixer;

const vo\_functions\_t \*video\_out;

// Frames buffered in the vo ready to flip. Currently always 0 or 1.

// This is really a vo variable but currently there's no suitable vo

// struct.

int num\_buffered\_frames;

// used to retry decoding after startup/seeking to compensate for codec delay

int startup\_decode\_retry;

// how long until we need to display the "current" frame

float time\_frame;

// AV sync: the next frame should be shown when the audio out has this

// much (in seconds) buffered data left. Increased when more data is

// written to the ao, decreased when moving to the next frame.

// In the audio-only case used as a timer since the last seek

// by the audio CPU usage meter.

double delay;

float begin\_skip; ///< start time of the current skip while on edlout mode

// audio is muted if either EDL or user activates mute

short edl\_muted; ///< Stores whether EDL is currently in muted mode.

short user\_muted; ///< Stores whether user wanted muted mode.

int global\_sub\_size; // this encompasses all subtitle sources

int global\_sub\_pos; // this encompasses all subtitle sources

int set\_of\_sub\_pos;

int set\_of\_sub\_size;

int sub\_counts[SUB\_SOURCES];

#ifdef CONFIG\_ASS

// set\_of\_ass\_tracks[i] contains subtitles from set\_of\_subtitles[i]

// parsed by libass or NULL if format unsupported

ASS\_Track\* set\_of\_ass\_tracks[MAX\_SUBTITLE\_FILES];

#endif

sub\_data\* set\_of\_subtitles[MAX\_SUBTITLE\_FILES];

int **file\_format**;

#ifdef CONFIG\_DVBIN

int last\_dvb\_step;

int dvbin\_reopen;

#endif

int was\_paused;

#ifdef CONFIG\_DVDNAV

struct mp\_image \*nav\_smpi; ///< last decoded dvdnav video image

unsigned char \*nav\_buffer; ///< last read dvdnav video frame

unsigned char \*nav\_start; ///< pointer to last read video buffer

int nav\_in\_size; ///< last read size

#endif

} **MPContext**;

**static MPContext \*mpctx = &mpctx\_s;**

static MPContext **mpctx\_s** = {

.osd\_function = OSD\_PLAY,

.begin\_skip = MP\_NOPTS\_VALUE,

.play\_tree\_step = 1,

.global\_sub\_pos = -1,

.set\_of\_sub\_pos = -1,

.file\_format = DEMUXER\_TYPE\_UNKNOWN,

.loop\_times = -1,

#ifdef CONFIG\_DVBIN

.last\_dvb\_step = 1,

#endif

};

typedef struct **ao\_functions\_s**

{

const ao\_info\_t \*info;

int (\*control)(int cmd,void \*arg);

int (\*init)(int rate,int channels,int format,int flags);

void (\*uninit)(int immed);

void (\*reset)(void);

int (\*get\_space)(void);

int (\*play)(void\* data,int len,int flags);

float (\*get\_delay)(void);

void (\*pause)(void);

void (\*resume)(void);

} **ao\_functions\_t**;

/\* global data used by mplayer and plugins \*/

typedef struct **ao\_data** {

int samplerate;

int channels;

int format;

int bps;

int outburst;

int buffersize;

int pts;

} **ao\_data\_t**;

typedef struct **stream\_info\_st** {

const char \*info;

const char \*name;

const char \*author;

const char \*comment;

/// mode isn't used atm (ie always READ) but it shouldn't be ignored

/// opts is at least in it's defaults settings and may have been

/// altered by url parsing if enabled and the options string parsing.

int (\*open)(struct stream\* st, int mode, void\* opts, int\* file\_format);

const char\* protocols[MAX\_STREAM\_PROTOCOLS];

const void\* opts;

int opts\_url; /\* If this is 1 we will parse the url as an option string

\* too. Otherwise options are only parsed from the

\* options string given to open\_stream\_plugin \*/

} **stream\_info\_t**;

typedef struct **stream** {

// Read

int (\*fill\_buffer)(struct stream \*s, char\* buffer, int max\_len);

// Write

int (\*write\_buffer)(struct stream \*s, char\* buffer, int len);

// Seek

int (\*seek)(struct stream \*s,off\_t pos);

// Control

// Will be later used to let streams like dvd and cdda report

// their structure (ie tracks, chapters, etc)

int (\*control)(struct stream \*s,int cmd,void\* arg);

// Close

void (\*close)(struct stream \*s);

int fd; // file descriptor, see man open(2)

int type; // see STREAMTYPE\_\*

int flags;

int sector\_size; // sector size (seek will be aligned on this size if non 0)

int read\_chunk; // maximum amount of data to read at once to limit latency (0 for default)

unsigned int buf\_pos,buf\_len;

off\_t pos,start\_pos,end\_pos;

int eof;

int mode; //STREAM\_READ or STREAM\_WRITE

unsigned int cache\_pid;

void\* cache\_data;

void\* priv; // used for DVD, TV, RTSP etc

char\* url; // strdup() of filename/url

#ifdef CONFIG\_NETWORKING

streaming\_ctrl\_t \***streaming\_ctrl**;

#endif

unsigned char buffer[STREAM\_BUFFER\_SIZE>STREAM\_MAX\_SECTOR\_SIZE?STREAM\_BUFFER\_SIZE:STREAM\_MAX\_SECTOR\_SIZE];

FILE \*capture\_file;

} **stream\_t**;

typedef struct **demuxer {**

const **demuxer\_desc\_t** \***desc**; ///< Demuxer description structure

off\_t filepos; // input stream current pos.

off\_t movi\_start;

off\_t movi\_end;

stream\_t \***stream**;

double stream\_pts; // current stream pts, if applicable (e.g. dvd)

double reference\_clock;

char \*filename; ///< Needed by avs\_check\_file

int synced; // stream synced (used by mpeg)

int type; // demuxer type: mpeg PS, mpeg ES, avi, avi-ni, avi-nini, asf

int **file\_format**; // file format: mpeg/avi/asf

int seekable; // flag

//

**demux\_stream\_t** \***audio**; // audio buffer/demuxer

demux\_stream\_t \*video; // video buffer/demuxer

demux\_stream\_t \*sub; // dvd subtitle buffer/demuxer

// stream headers:

void\* **a\_streams**[MAX\_A\_STREAMS]; // audio streams (sh\_audio\_t)

void\* v\_streams[MAX\_V\_STREAMS]; // video streams (sh\_video\_t)

void \*s\_streams[MAX\_S\_STREAMS]; // dvd subtitles (flag)

// pointer to teletext decoder private data, if demuxer stream contains teletext

void \*teletext;

int num\_titles;

demux\_chapter\_t\* chapters;

int num\_chapters;

demux\_attachment\_t\* attachments;

int num\_attachments;

void\* priv; // fileformat-dependent data

char\*\* info;

**} demuxer\_t;**

typedef struct {

int buffer\_pos; // current buffer position

int buffer\_size; // current buffer size

unsigned char\* buffer; // current buffer, never free() it, always use free\_demux\_packet(buffer\_ref);

double pts; // current buffer's pts

int pts\_bytes; // number of bytes read after last pts stamp

int eof; // end of demuxed stream? (true if all buffer empty)

off\_t pos; // position in the input stream (file)

off\_t dpos; // position in the demuxed stream

int pack\_no; // serial number of packet

int flags; // flags of current packet (keyframe etc)

int non\_interleaved; // 1 if this stream is not properly interleaved,

// so e.g. subtitle handling must do explicit reads.

//---------------

int packs; // number of packets in buffer

int bytes; // total bytes of packets in buffer

demux\_packet\_t \*first; // read to first buffer after the current buffer from here

demux\_packet\_t \*last; // append new packets from input stream to here

demux\_packet\_t \*current;// needed for refcounting of the buffer

int id; // stream ID (for multiple audio/video streams)

struct demuxer \*demuxer; // parent demuxer structure (stream handler)

// ---- asf -----

demux\_packet\_t \*asf\_packet; // read asf fragments here

int asf\_seq;

// ---- mov -----

unsigned int ss\_mul,ss\_div;

// ---- stream header ----

void\* sh;

} **demux\_stream\_t**;

#define **SH\_COMMON** \

**demux\_stream\_t \*ds**; \

struct codecs \*codec; \

unsigned int format; \

int initialized; \

float stream\_delay; /\* number of seconds stream should be delayed (according to dwStart or similar) \*/ \

/\* things needed for parsing \*/ \

int needs\_parsing; \

struct AVCodecContext \*avctx; \

struct AVCodecParserContext \*parser; \

/\* audio: last known pts value in output from decoder \

\* video: predicted/interpolated PTS of the current frame \*/ \

double pts; \

/\* codec-specific: \*/ \

void\* context; /\* codec-specific stuff (usually HANDLE or struct pointer) \*/ \

char\* lang; /\* track language \*/ \

int default\_track; \

typedef struct sh\_common {

SH\_COMMON

} **sh\_common\_t**;

typedef struct **sh\_audio** {

**SH\_COMMON**

int aid;

// **output format**:

int sample\_format;

int samplerate;

int samplesize;

int channels;

int o\_bps; // == samplerate\*samplesize\*channels (uncompr. bytes/sec)

int i\_bps; // == bitrate (compressed bytes/sec)

// **in buffers**:

int audio\_in\_minsize; // max. compressed packet size (== min. in buffer size)

char\* a\_in\_buffer;

int a\_in\_buffer\_len;

int a\_in\_buffer\_size;

// **decoder buffers**:

int audio\_out\_minsize; // max. uncompressed packet size (==min. out buffsize)

char\* a\_buffer;

int a\_buffer\_len;

int a\_buffer\_size;

// **output buffers**:

char\* a\_out\_buffer;

int a\_out\_buffer\_len;

int a\_out\_buffer\_size;

// void\* **audio\_out**; // the audio\_out handle, used for this audio stream

struct af\_stream \*afilter; // the audio filter stream

const struct ad\_functions \*ad\_driver;

#ifdef CONFIG\_DYNAMIC\_PLUGINS

void \*dec\_handle;

#endif

// win32-compatible codec parameters:

AVIStreamHeader audio;

WAVEFORMATEX\* wf;

// codec-specific:

unsigned char\* codecdata; // extra header data passed from demuxer to codec

int codecdata\_len;

int pts\_bytes; // bytes output by decoder after last known pts

} **sh\_audio\_t**;