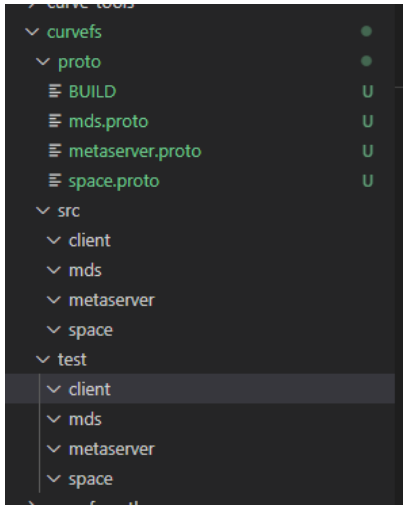
curve文件系统元数据proto（代码接口定义，已实现）

# 1、代码结构和代码目录

curve文件系统是相对于curve块设备比较独立的一块，在当前curve项目的目录下，增加一个一级目录curvefs，curvefs下有自己独立的proto\src\test。




# 2、文件系统proto定义

## 2.1 mds.proto

mds.proto

```
/*
 *  Copyright (c) 2020 NetEase Inc.
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
```

```
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

syntax="proto2";
package curvefs.mds;
option cc_generic_services = true;

enum FSStatusCode {
    OK = 0;             //
    UNKNOWN_ERROR = 1;  //
    NOSPACE = 2;        //
}

// fs interface
message GetFsInfoRequest {
    optional uint32 fsId = 1;     // fs id
    optional string fsName = 2;   // fs name
}

message mountPoint {
    required string host = 1;
    required string mountDir = 2;     //
}
message Volume {
    required uint64 volumeSize = 1;
    required uint64 blockSize = 2;
    required string volumeName = 3;
    required string user = 4;
    optional string password = 5;
}

message FsInfo {
    required uint32 fsId = 1;
```

```
    required string fsName = 2;
    required uint64 rootInodeId = 3;
    required uint64 capacity = 4;
    required uint64 blockSize = 5;
    required Volume volume = 6;
    required uint32 mountNum = 7;
    repeated mountPoint mountpoints = 8;
}

message GetFsInfoResponse {
    required MetaStatusCode statusCode = 1;
    optional FsInfo fsInfo = 2;
}

message CreateFsRequest {
    required string fsName = 1;
    required uint64 blockSize = 2;
    required Volume volume = 3;
}

message CreateFsResponse {
    required MetaStatusCode statusCode = 1;
    optional FsInfo fsInfo = 2;
}

message MountFsRequest {
    required string fsName = 1;
    required mountPoint mountpoint = 2;
}

message MountFsResponse {
    required MetaStatusCode statusCode = 1;
    optional FsInfo fsInfo = 2;
}

message UmountFsRequest {
    required string fsName = 1;
    required mountPoint mountpoint = 2;
```

```
}

message UmountFsResponse {
    required MetaStatusCode statusCode = 1;
}

/* UpdateFsInfoRequest
message UpdateFsInfoRequest {
    required string fsName = 1;
    //
}

message UpdateFsInfoResponse {
    required MetaStatusCode statusCode = 1;
    optional FsInfo fsInfo = 2;
}
*/

message DeleteFsInfoRequest {
    required string fsName = 2;
}

message DeleteFsInfoResponse {
    required MetaStatusCode statusCode = 1;
}

service MdsService {
    // fs interface
    rpc CreateFs(CreateFsRequest) returns (CreateFsResponse);
    rpc MountFs(MountFsRequest) returns (MountFsResponse);
    rpc UmountFs(UmountFsRequest) returns (UmountFsResponse);
    rpc GetFsInfo(GetFsInfoRequest) returns (GetFsInfoResponse);
```

```
    // rpc UpdateFsInfo(UpdateFsInfoRequest) returns (UpdateFsInfoResponse);
    rpc DeleteFsInfo(DeleteFsInfoRequest) returns (DeleteFsInfoResponse);
}
```

## 2.2 metaserver.proto

metaserver.proto

```
/*
 *  Copyright (c) 2020 NetEase Inc.
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

syntax="proto2";
package curvefs.metaserver;
option cc_generic_services = true;

enum MetaStatusCode {
    OK = 0;             //
    UNKNOWN_ERROR = 1;  //
    NOSPACE = 2;        //
}

// dentry interface
```

```
message GetDentryRequest {
    required uint32 fsId = 1;
    required uint64 parentInodeId = 2;
    required string name = 3;
}

message Dentry {
    required uint32 fsId = 1;
    required uint64 inodeId = 2;
    required uint64 parentInodeId = 3;
    required string name = 4;
}

message GetDentryResponse {
    required MetaStatusCode statusCode = 1;
    optional Dentry dentry = 2;
}

message ListDentryRequest {
    required uint32 fsId = 1;
    required uint64 dirInodeId = 2;
    optional uint64 last = 3;   //
    optional uint64 count = 4;    //
}

message ListDentryResponse {
    required MetaStatusCode statusCode = 1;
    repeated Dentry dentrys = 2;
}

message CreateDentryRequest {
    required Dentry dentry = 1;
}

message CreateDentryResponse {
    required MetaStatusCode statusCode = 1;
}
```

```
message UpdateDentryRequest {
    required Dentry dentry = 1;
}

message UpdateDentryResponse {
    required MetaStatusCode statusCode = 1;
}

message DeleteDentryRequest {
    required uint32 fsId = 1;
    required uint64 parentInodeId = 2;
    required string name = 3;
}

message DeleteDentryResponse {
    required MetaStatusCode statusCode = 1;
}

// inode interface
message GetInodeRequest {
    required uint32 fsId = 1;
    required uint64 inodeId = 2;
}

enum FileType {
    TYPE_DIRECTORY = 1;
    TYPE_FILE = 2;
    TYPE_SYM_LINK = 3;
};

message VolumeExtent {
    required uint64 fsOffset = 1;
    required uint64 volumeOffset = 2;
    required uint64 length = 3;
    required bool isused = 4;
}

message Inode {
```

```
    required uint64 inodeId = 1;
    required uint32 fsId = 2;
    required uint64 length = 3;
    required uint32 ctime = 4;
    required uint32 mtime = 5;
    required uint32 atime = 6;
    required uint32 uid = 7;
    required uint32 gid = 8;
    required uint32 mode = 9;
    required sint32 nlink = 10;
    required FileType type = 11;
    optional string symlink = 12;   // TYPE_SYM_LINK only
    repeated VolumeExtent volumeExtents = 13;  // TYPE_FILE only
}

message GetInodeResponse {
    required MetaStatusCode statusCode = 1;
    optional Inode inode = 2;
}

message CreateInodeRequest {
    required uint32 fsId = 1;
    required uint64 length = 2;
    required uint32 uid = 3;
    required uint32 gid = 4;
    required uint32 mode = 5;
    required FileType type = 6;
    optional string symlink = 7;   // TYPE_SYM_LINK only
}

message CreateInodeResponse {
    required MetaStatusCode statusCode = 1;
    optional Inode inode = 2;
}

message VolumeExtentList {
    repeated VolumeExtent volumeExtents = 1;
}
```

```
message UpdateInodeRequest {
    required uint64 inodeId = 1;
    required uint32 fsId = 2;
    optional uint64 length = 3;
    optional uint32 ctime = 4;
    optional uint32 mtime = 5;
    optional uint32 atime = 6;
    optional uint32 uid = 7;
    optional uint32 gid = 8;
    optional uint32 mode = 9;
    optional VolumeExtentList volumeExtentList = 10;  // TYPE_FILE only
}

message UpdateInodeResponse {
    required MetaStatusCode statusCode = 1;
}
message DeleteInodeRequest {
    required uint32 fsId = 1;
    required uint64 inodeId = 2;
}

message DeleteInodeResponse {
    required MetaStatusCode statusCode = 1;
}

service MetaServerService {
    // dentry interface
    rpc GetDentry(GetDentryRequest) returns (GetDentryResponse);
    rpc ListDentry(ListDentryRequest) returns (ListDentryResponse);
    rpc CreateDentry(CreateDentryRequest) returns (CreateDentryResponse);
    rpc UpdateDentry(UpdateDentryRequest) returns (UpdateDentryResponse);
    rpc DeleteDentry(DeleteDentryRequest) returns (DeleteDentryResponse);

    // inode interface
    rpc GetInode(GetInodeRequest) returns (GetInodeResponse);
    rpc CreateInode(CreateInodeRequest) returns (CreateInodeResponse);
    rpc UpdateInode(UpdateInodeRequest) returns (UpdateInodeResponse);
    rpc DeleteInode(DeleteInodeRequest) returns (DeleteInodeResponse);
```

```
}
```

## 2.3 space.proto

space.proto

```
/*
 *  Copyright (c) 2020 NetEase Inc.
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

syntax="proto2";
package curvefs.space;
option cc_generic_services = true;

enum SpaceStatusCode {
    OK = 0;             //
    UNKNOWN_ERROR = 1;  //
    NOSPACE = 2;        //
}

// space interface
message Extent {
```

```
    required uint64 offset = 1;  //
    required uint32 length = 2;  //
}

enum AllocateType {
    NONE = 0;
    SMALL = 1;     //
    BIG = 2;       //
}

message AllocateHint {
    optional AllocateType allocType = 1;   //
    optional uint64 leftOffset = 2;        //
    optional uint64 rightOffset = 3;       //
}

message InitSpaceRequest {
    required uint32 fsId = 1;
    required Volume volumeInfo = 2;
}

message InitSpaceResponse {
    required SpaceStatusCode status = 1;  //
}

message AllocateSpaceRequest {
    required uint32 fsId = 1;         // ID
    required uint32 size = 2;         //
    optional AllocateHint allocHint = 3;
}

message AllocateSpaceResponse {
    required SpaceStatusCode status = 1;  //
    repeated Extent extents = 2;     // repeated
}

message DeallocateSpaceRequest {
    required uint32 fsId = 1;
```

```
    repeated Extent extents = 2;     //
}

message DeallocateSpaceResponse {
    required SpaceStatusCode status = 1;
}

message StatSpaceRequest {
    required uint32 fsId = 1;
}

message StatSpaceResponse {
    required SpaceStatusCode status = 1;
    optional uint64 blockSize = 2;
    optional uint64 totalBlock = 3;
    optional uint64 availabalBlock = 4;
    optional uint64 usedBlock = 5;
}

message UnInitSpaceRequest {
    required uint32 fsId = 1;
}

message UnInitSpaceResponse {
    required SpaceStatusCode status = 1;
}

service SpaceAllocService {
    // space interface
    rpc InitSpace(InitSpaceRequest) returns (InitSpaceResponse);
    rpc AllocateSpace(AllocateSpaceRequest) returns (AllocateSpaceResponse);
    rpc DeallocateSpace(DeallocateSpaceRequest) returns (DeallocateSpaceResponse);
```

```
    rpc StatSpace(StatSpaceRequest) returns (StatSpaceResponse);
    rpc UnInitSpace(UnInitSpaceRequest) returns (UnInitSpaceResponse);
}
```