NagaeVFS 文档

ZRY

2024-03-19

API Ver. 1.1.0

1 版本说明

当前版本提供以下功能。

1.1 VFS

一个可挂载的虚拟文件系统。其主机接口兼容 afero.Fs。

1.2 INIT WASM

用于初始化文件系统挂载的 WASM 虚拟机。

该虚拟机提供以下接口：
1. 在主机上创建和释放 afero.Fs 对象的接口。
2. 挂载主机上被创建的 afero.Fs 对象的接口。
3. 在主机 VFS 上创建目录的接口。
4. 读取配置文件中 ExKV 键值对内容的接口。

当前版本可创建的 afero.Fs 对象，由 afero 包本身提供的有：

- afero.OsFs
- afero.MemMapFs
- afero.BasePathFs
- afero.RegexpFs
- afero.ReadOnlyFs
- afero.CopyOnWriteFs
- afero.CacheOnReadFs

其中经过简单测试并通过的有：

- afero.OsFs
- afero.MemMapFs
- afero.BasePathFs
- afero.ReadOnlyFs

其中经过测试有问题的：

- afero.RegexpFs
  - Readdir 功能在遇到无法匹配的文件时产生 error

    该问题来自 afero.RegexpFs 本身，故不提供其修正。

    谨慎考虑使用 RegexpFs。

当前版本暂未提供其他 afero.Fs 对象支持。

2 INIT WASI 接口

2.1 WASM 导出函数

2.1.1.1 函数 auto_mount

名称	auto_mount			
说明	执行自动挂载，即 WASM 的实际功能入口			
原型	`fn auto_mount() -> (exit_code: i32);`			
返回值	名称	WASM 类型	语义类型	说明
	exit_code	i32	i32	挂载结果 值 | 说明 0 | 挂载成功 !=0 | 挂载失败

2.1.1.2 函数 _start

名称	_start
说明	WASI 的主入口
原型	`fn _start();`

2.2 主机导出模块

2.2.1 模块 ngvfs_init

2.2.1.1 函数 api_ver_get

名称	api_ver_get			
说明	获得 NGVFS API 版本			
原型	`fn api_ver_get(ptrMajorVersion: i32,ptrMinorVersion: i32,ptrPatchVersion: i32);`			
参数	名称	WASM 类型	语义类型	说明
	ptrMajorVersion	i32	*mut i32	存放获取到的 MajorVersion 的 i32 变量指针
	ptrMinorVersion	i32	*mut i32	存放获取到的 MinorVersion 的 i32 变量指针
	ptrPatchVersion	i32	*mut i32	存放获取到的 PatchVersion 的 i32 变量指针

2.2.1.2 函数 exkv_get_len

名称	exkv_get_len			
说明	通过 key 获得 exkv 中 value 的长度			
原型	`fn exkv_get_len(ptrKey: i32,lenKey: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	ptrKey	i32	*const u8	key 的指针
	lenKey	i32	usize	key 的长度
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -2 / key 不存在 -1 / 读取 key 时内存越界 >=0 / 返回值即为 value 的长度

2.2.1.3 函数 `exkv_get_val`

名称	exkv_get_val			
说明	通过 key 获得 exkv 中的 value			
原型	`fn exkv_get_val(ptrKey: i32,lenKey: i32,ptrBuf: i32,lenBuf: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	ptrKey	i32	*const u8	key 的指针
	lenKey	i32	usize	key 的长度
	ptrBuf	i32	*mut u8	存放值的缓冲区指针
	lenBuf	i32	usize	缓冲区长度。缓冲区不足则值会被截断
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -3 / 写入 buf 时内存越界 -2 / key 不存在 -1 / 读取 key 时内存越界 0 / 缓冲区长度小于等于 0 >0 / 写入缓冲区的值的长度

2.2.1.4 函数 `vfs_mount`

<table>
<tr><td>名称</td><td colspan="4">vfs_mount</td></tr>
<tr><td>说明</td><td colspan="4">挂载指定句柄的 afs 文件系统实例至指定挂载点</td></tr>
<tr><td>原型</td><td colspan="4"><code>fn vfs_mount(hAfs: i32,ptrMntPath: i32,lenMntPath: i32) -> (result: i32);</code></td></tr>
<tr><td rowspan="4">参数</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td>hAfs</td><td>i32</td><td>i32</td><td>afs 实例句柄</td></tr>
<tr><td>ptrMntPath</td><td>i32</td><td>*const u8</td><td>挂载点路径的指针</td></tr>
<tr><td>lenMntPath</td><td>i32</td><td>usize</td><td>挂载点路径的长度</td></tr>
<tr><td rowspan="2">返回值</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td>result</td><td>i32</td><td>enum</td><td>结果
<table><tr><td>值</td><td>说明</td></tr><tr><td>-4</td><td>挂载时发生内部错误，详见日志</td></tr><tr><td>-3</td><td>挂载点已经被挂载</td></tr><tr><td>-2</td><td>读取挂载点路径时内存越界</td></tr><tr><td>-1</td><td>afs 句柄无效</td></tr><tr><td>0</td><td>挂载成功</td></tr></table></td></tr>
</table>

2.2.1.5 函数 `vfs_mkdir`

<table>
<tr><td>名称</td><td colspan="4">vfs_mkdir</td></tr>
<tr><td>说明</td><td colspan="4">创建目录，类似 mkdir -p，用于准备挂载点</td></tr>
<tr><td>原型</td><td colspan="4"><code>fn vfs_mkdir(ptrPath: i32,lenPath: i32,mode: i32) -> (result: i32);</code></td></tr>
<tr><td rowspan="4">参数</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td>ptrPath</td><td>i32</td><td>*const u8</td><td>目录路径的指针</td></tr>
<tr><td>lenPath</td><td>i32</td><td>usize</td><td>目录路径的长度</td></tr>
<tr><td>mode</td><td>i32</td><td>i32</td><td>目录的权限模式</td></tr>
<tr><td rowspan="2">返回值</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td>result</td><td>i32</td><td>enum</td><td>结果
<table><tr><td>值</td><td>说明</td></tr><tr><td>-2</td><td>创建目录失败，详见日志</td></tr><tr><td>-1</td><td>读取路径时内存越界</td></tr><tr><td>0</td><td>创建成功</td></tr></table></td></tr>
</table>

2.2.1.6 函数 `afs_free`

名称	afs_free			
说明	释放指定句柄的 afs 实例			
原型	`fn afs_free(hAfs: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hAfs	i32	i32	afs 实例句柄
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 | 说明 -1 | 句柄不存在 0 | 释放成功

2.2.1.7 函数 `afs_mkdir`

名称	afs_mkdir			
说明	在 afero.Fs 实例上创建目录，类似 mkdir -p，用于准备挂载点，但是是在 afero.Fs 实例上进行。			
原型	`fn afs_mkdir(hAfs: i32,ptrPath: i32,lenPath: i32,mode: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hAfs	i32	i32	afs 实例句柄
	ptrPath	i32	*const u8	Path 的指针
	lenPath	i32	usize	Path 的长度
	mode	i32	i32	目录的权限模式
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 | 说明 -3 | 读取 Path 时内存越界 -2 | afs 实例句柄无效 -1 | 创建目录失败，详见日志 0 | 创建目录成功

2.2.1.8 函数 `afs_create_osfs`

名称	afs_create_osfs
说明	创建 afero.OsFs 实例
原型	`fn afs_create_osfs() -> (result: i32);`

返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 | 说明 -1 | 申请 afs 句柄资源失败 >0 | 创建的 afs 实例的句柄

2.2.1.9 函数 `afs_create_memfs`

名称	afs_create_memfs			
说明	创建 afero.MemMapFs 实例			
原型	`fn afs_create_memfs() -> (result: i32);`			
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 | 说明 -1 | 申请 afs 句柄资源失败 >0 | 创建的 afs 实例的句柄

2.2.1.10 函数 `afs_create_bpfs`

名称	afs_create_bpfs			
说明	创建 afero.BasePathFs 实例			
原型	`fn afs_create_bpfs(hBaseAfs: i32,ptrBasePath: i32,lenBasePath: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hBaseAfs	i32	i32	基于的 afs 实例句柄
	ptrBasePath	i32	*const u8	BasePath 的指针
	lenBasePath	i32	usize	BasePath 的长度

返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -3 读取 BasePath 时内存越界 -2 afs 实例句柄无效 -1 申请 afs 句柄资源失败 >0 创建的 afs 实例的句柄

2.2.1.11 函数 `afs_create_regfs`

名称	afs_create_regfs			
说明	创建 afero.RegexpFs 实例，使用字符串形式的 golang 正则表达式			
原型	`fn afs_create_regfs(hBaseAfs: i32,ptrRegExp: i32,lenRegExp: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hBaseAfs	i32	i32	基于的 afs 实例句柄
	ptrRegExp	i32	*const u8	RegExp 的指针
	lenRegExp	i32	usize	RegExp 的长度
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -4 编译正则表达式失败 -3 读取 RegExp 时内存越界 -2 afs 实例句柄无效 -1 申请 afs 句柄资源失败 >0 创建的 afs 实例的句柄

2.2.1.12 函数 `afs_create_rofs`

名称	afs_create_rofs
说明	创建 afero.ReadOnlyFs 实例

原型	`fn afs_create_rofs(hBaseAfs: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hBaseAfs	i32	i32	基于的 afs 实例句柄
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -2 / afs 实例句柄无效 -1 / 申请 afs 句柄资源失败 >0 / 创建的 afs 实例的句柄

2.2.1.13 函数 `afs_create_cowfs`

名称	afs_create_cowfs			
说明	创建 afero.CopyOnWriteFs 实例			
原型	`fn afs_create_cowfs(hRoAfs: i32,hWrAfs: i32) -> (result: i32);`			
参数	名称	WASM 类型	语义类型	说明
	hRoAfs	i32	i32	基于的只读 afs 实例句柄
	hWrAfs	i32	i32	基于的可写 afs 实例句柄
返回值	名称	WASM 类型	语义类型	说明
	result	i32	enum	结果 值 / 说明 -3 / 可写 afs 实例句柄无效 -2 / 只读 afs 实例句柄无效 -1 / 申请 afs 句柄资源失败 >0 / 创建的 afs 实例的句柄

2.2.1.14 函数 `afs_create_corfs`

名称	afs_create_corfs
说明	创建 afero.CacheOnReadFs 实例
原型	`fn afs_create_corfs(hRoAfs: i32,hWrAfs: i32,cacheTime: i32) -> (result: i32);`

<table>
<tr><td>参数</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td></td><td>hRoAfs</td><td>i32</td><td>i32</td><td>基于的 afs 实例句柄</td></tr>
<tr><td></td><td>hWrAfs</td><td>i32</td><td>i32</td><td>缓存 afs 实例句柄</td></tr>
<tr><td></td><td>cacheTime</td><td>i32</td><td>i32</td><td>缓存时间，单位为秒，0 为永久缓存</td></tr>
<tr><td>返回值</td><td>名称</td><td>WASM 类型</td><td>语义类型</td><td>说明</td></tr>
<tr><td></td><td>result</td><td>i32</td><td>enum</td><td>结果<table><tr><td>值</td><td>说明</td></tr><tr><td>-4</td><td>缓存时间小于 0</td></tr><tr><td>-3</td><td>可写 afs 实例句柄无效</td></tr><tr><td>-2</td><td>只读 afs 实例句柄无效</td></tr><tr><td>-1</td><td>申请 afs 句柄资源失败</td></tr><tr><td>>0</td><td>创建的 afs 实例的句柄</td></tr></table></td></tr>
</table>

2.3 WASM 侧 Rust Crate

提供有用于 WASM 侧的接口包装 Rust Crate。

目前本文档内暂不提供该 Crate 的文档。

该包的交互式文档可使用 rustdoc 生成:

```
cd init-wasm
just doc_html
```

生成的文档在 `init-wasm/ngvfs_init_wasm_lib/target/doc/ngvfs_init_wasm_lib`

若安装有 `SimpleHttpTestServer`，可继续使用如下命令:

```
# 端口号可换成其他值
just serv_doc "localhost:8088"
```

然后使用浏览器打开 http://localhsot:8088/。