
Hallyd

Release 0.9

unknown

Feb 16, 2025

CONTENTS

1	License	3
2	Up-To-Date?	5
3	User Manual	7
3.1	Getting Started	7
4	API Reference	9
4.1	hallyd package	9
	Python Module Index	73
	Index	75

Hallyd is a library that contains all kinds of tools for various tasks around filesystem, and other system features. This is intended to be used by other projects available here. External developers should double-check if it actually suffices their needs!

LICENSE

Hallyd is distributed under the terms of the AGPL 3 license. This also affects all included files without a license header (non-source files like images), unless they are explicitly mentioned as third-party content. Read the ‘Dependencies’ section for included third-party stuff.

UP-TO-DATE?

Are you currently reading from another source than the homepage? If you are in doubt whether your package is up-to-date, you should visit the project homepage and check that. You are currently reading the documentation for version 0.9.

USER MANUAL

3.1 Getting Started

3.1.1 Foo

This section is about getting ready.

API REFERENCE

4.1 hallyd package

4.1.1 Subpackages

hallyd._aux namespace

Submodules

hallyd._aux.services_helper__call_action module

`hallyd._aux.services_helper__call_action.main(token)`

Parameters

token (*str*) –

Return type

None

hallyd._aux.subprocess_helper__call_function module

`hallyd._aux.subprocess_helper__call_function.main()`

4.1.2 Submodules

4.1.3 hallyd.bindle module

exception `hallyd.bindle.SerializingError`

Bases: `ValueError`

`hallyd.bindle._by_qualname(module, qualname)`

`hallyd.bindle._deserialize_object_json_loads_object_hook(dict_)`

`hallyd.bindle._enum(enum_type, name)`

`hallyd.bundle._filter_unneeded_dict_entries(obj_type, obj_dict)`

Parameters

`obj_dict` (*dict*) –

Return type

dict

`hallyd.bundle._new_bytes(b64str)`

`hallyd.bundle._new_datetime_datetime(timestamp)`

`hallyd.bundle._new_datetime_timedelta(seconds)`

`hallyd.bundle._new_functools_partial(func, args, kwargs)`

`hallyd.bundle._new_set(items)`

`hallyd.bundle._serialize_object_json_dumps_default(obj)`

`hallyd.bundle.dict_from_object(obj)`

Parameters

`obj` (*Optional[Any]*) –

Return type

Optional[dict[str, Optional[Any]]]

`hallyd.bundle.dump(obj, fp)`

Serialize an object to a file object. See also `load()`.

Parameters

- `obj` (*Optional[Any]*) –
- `fp` (*IO*) –

Return type

None

`hallyd.bundle.dumps(obj)`

Serialize an object to a string. See also `loads()`.

Parameters

`obj` (*Optional[Any]*) –

Return type

str

`hallyd.bundle.load(fp)`

Deserialize an object from a file object. See also `dump()` and `loads()`.

Parameters

`fp` (*IO*) –

Return type

Optional[Any]

`hallyd.bundle.loads(sj)`

Deserialize an object from a string. See also `dumps()` and `load()`.

Parameters

`sj` (*AnyStr*) –

Return type*Optional[Any]*

4.1.4 hallyd.cleanup module

Cleanup tasks can do arbitrary things after your process is terminated, like removing temporary files or other system resources that are not cleaned up automatically by the operating system.

class hallyd.cleanup._CleanupTask(*task_id*)

Bases: object

Parameters

task_id (*Union[str, Path]*) –

Id

alias of str

remove()

Return type

None

property task_id: str

hallyd.cleanup._add_cleanup_task(*func, args, kwargs*)

Parameters

- **func** (*Union[Callable, SupportsQualifiedName]*) –
- **args** (*tuple*) –
- **kwargs** (*dict*) –

Return type

_CleanupTask

hallyd.cleanup._current_cleanup_scope()

Return type

str

hallyd.cleanup._do_cleanup()

Return type

None

hallyd.cleanup._guarded_cleanup(*process_permanent_id*)

Parameters

process_permanent_id (*str*) –

Return type

None

hallyd.cleanup._have_same_fs_root(*p1, p2*)

Parameters

- **p1** (*str*) –
- **p2** (*str*) –

Return type

bool

Add a cleanup task to be executed when the current process is terminated, and return a task controller object that allows early execution and other things. This will run in a separate process, once the current process is fully terminated.

Parameters

- **func** (*Union[Callable, SupportsQualifiedName]*) – The function to call. Must not be a (non-static) method of some object.
- **args** – The function call args.
- **kwargs** – The function call kwargs.

Return type

_CleanupTask

Make sure that cleanup will take place once the current process is terminated. You usually do not need to call it.

Return the task controller object by task id.

Parameters
task_id (*str*) – The task id.

Return type

_CleanupTask

Mark the current process as cleanup scope, so even for child processes, cleanup will not happen before this process is terminated.

Return type

None

4.1.5 hallyd.coding module

class hallyd.coding.Editor(srcfile)

Bases: object

Parameters
srcfile (*Path*) –

class _ClassHandle(editor, class_name)

Bases: *_Handle*, *_WithCodePositionsHandleMixin*, *_WithDecorationSupportHandleMixin*, *_RemovableHandleMixin*

_abc_impl = <_abc._abc_data object>

property _ast: Optional[ClassDef]

add_method(body)

Parameters
body (*str*) –


```

    method_by_name(method_name)

    property methods

    property name: str

class _FunctionHandle(editor, function_name, class_handle=None)
    Bases: _Handle, _WithCodePositionsHandleMixin, _WithDecorationSupportHandleMixin,
    _RemovableHandleMixin

    Parameters
        • editor (Editor) –
        • function_name (str) –

    _abc_impl = <_abc._abc_data object>

    property _ast: Optional[FunctionDef]

    property name: str

class _Handle(editor)
    Bases: ABC

    _abc_impl = <_abc._abc_data object>

    abstract property _ast: Optional[AST]

    property _editor

class _RemovableHandleMixin
    Bases: object

    remove()

class _WithCodePositionsHandleMixin
    Bases: object

    property ends_at_position

    property starts_at_position

class _WithDecorationSupportHandleMixin
    Bases: object

    _WithDecorationSupportHandleMixin__decoration_ast_to_code_indexes(decoration_ast)

    add_decoration(code)

    Parameters
        code (str) –

    property decorations

    remove_decoration(idx)

__indentation_for_code_fragment()

    Parameters
        code (str) –

```

Return type

str

property _ast: AST**_fix_body**(body)**Parameters****body** (str) –**Return type**

str

static _function_name_from_body(body)**add_class**(class_name, *, derived_from=None, docstring='TODO add some documentation here')**add_function**(body)**add_import**(module_name)**Parameters****module_name** (str) –**class_by_name**(class_name)**Parameters****class_name** (str) –**Return type**[_ClassHandle](#)**property code:** str**property indentation:** str**property path:** [Path](#)

4.1.6 hallyd.disk module

class hallyd.disk.Disk(dev_path)

Bases: object

Parameters**dev_path** ([Path](#)) –**__idpath**()**Return type***Optional*[str]**__lsblk**()**Return type**dict[str, *Any*]**__udev_property**(name)**Parameters****name** (str) –

```

    Return type
        Optional[str]
    _has_path(path)

    Parameters
        path (str) –

    Return type
        bool
    property is_disk: bool
    property is_removable: bool
    partition(part_no)

    Parameters
        part_no (int) –

    Return type
        DiskPartition
    partition_path(part_no)

    Parameters
        part_no (int) –

    Return type
        Path
    property partitions: list[hallyd.disk.DiskPartition]
    property path: Path
    property size_bytes: int
    property stable_path: Optional[str]
    stable_udev_filter()

    Return type
        str
class hallyd.disk.DiskIntent(disk, setup)
    Bases: object
    Parameters
        • disk (Disk) –
        • setup (DiskSetup) –
    __reread_partition_table()
    __write_partition_table(repartitionspecs)
    property disk
    repartition()

    Return type
        None

```

property setup

udev_rule_for_alias()

Return type

str

class hallyd.disk.DiskPartition(disk, part_no)

Bases: [Partition](#)

Parameters

- **disk** ([Disk](#)) –
- **part_no** (*int*) –

property disk

property fstype: Optional[[_PartitionType](#)]

property part_no

class hallyd.disk.DiskSetup(*partitions, identify_by, do_repartition=True, partition_table_type='gpt', name=None)

Bases: object

Parameters

- **path** – The path to the disk device file.
- **repartition** – If to write a new partition table (or reuse the existing one).
- **partitions** ([PartitionSetup](#)) –
- **identify_by** (*Iterable[str]*) –
- **do_repartition** (*bool*) –
- **partition_table_type** (*str*) –
- **name** (*Optional[str]*) –

[hallyd.disk.EfiPartitionSetup](#)(**kwargs)

class hallyd.disk.Mountpoint(partition, mountpoint, fstype)

Bases: object

Parameters

- **partition** ([Partition](#)) –
- **mountpoint** (*str*) –
- **fstype** ([_PartitionType](#)) –

fstab_line()

Return type

str

mount(*, prefix=None, create_before=True)

Parameters

- **prefix** (*Optional[str]*) –

```

        • create_before (bool) –

umount()

hallyd.disk.NotEfiPartitionSetup()

class hallyd.disk.OrderedPartitionSetupsEntry(part_no, partition_setup)
    Bases: object

    Parameters
        • part_no (int) –
        • partition_setup (PartitionSetup) –

    property part_no: int

    property partition_setup: Optional[PartitionSetup]

class hallyd.disk.Partition(path)
    Bases: object

    Parameters
        path (Path) –

    static by_partuuid(uuid_)

        Parameters
            uuid_ (str) –

        Return type
            Partition

    static by_uuid(uuid_)

        Parameters
            uuid_ (str) –

        Return type
            Partition

    property partuuid: Optional[str]

    property path: Path

    property stable_path: Path

    property uuid: Optional[str]

class hallyd.disk.PartitionSetup(*, index=None, fs_type=None, mountpoint=None, label=None,
do_format=True, use_in_raid=None, size=None, start_at_mb=None,
flag_bootable=False)

    Bases: _PartitionSetup

    Parameters
        • index (Optional[int]) – Partition index (counted from 1). Only needed in exotic cases.
        • use_in_raid (Optional[str]) – Name of the raid to use this partition for. See also the
            raid_name parameter of RaidPartitionSetup.__init__.
        • size (Optional[Union[int, Callable[[PartitionSizingEvent], int]]]) –
            The size in bytes.

```

- **start_at_mb** (*Optional[float]*) – The start offset in units of 1024^2 bytes.
- **flag_bootable** (*bool*) – If to flag this partition as bootable.
- **fs_type** (*Optional[_PartitionType]*) –
- **mountpoint** (*Optional[str]*) –
- **label** (*Optional[str]*) –
- **do_format** (*bool*) –

class hallyd.disk.PartitionSizingEvent(*disk_size*)

Bases: object

Parameters

disk_size (*int*) –

property **disk_size**: int

class hallyd.disk.PartitionTypes

Bases: object

EFI = hallyd.disk._PartitionType()

EXT4 = hallyd.disk._PartitionType()

RAID = hallyd.disk._PartitionType()

SWAP = hallyd.disk._PartitionType()

UNUSED = hallyd.disk._PartitionType()

class hallyd.disk.RaidPartition(*path*)

Bases: [Partition](#)

Parameters

path ([Path](#)) –

stop()

class hallyd.disk.RaidPartitionSetup(*raid_name*, *, *fs_type=None*, *mountpoint=None*, *label=None*,
do_format=True, *do_create_raid=True*)

Bases: [_PartitionSetup](#)

Parameters

- **raid_name** (*str*) – The raid to use for this partition. See also the `use_in_raid` parameter of `Partition.__init__`.
- **fs_type** (*Optional[_PartitionType]*) –
- **mountpoint** (*Optional[str]*) –
- **label** (*Optional[str]*) –
- **do_format** (*bool*) –
- **do_create_raid** (*bool*) –

class hallyd.disk.RaidSetup(*name*, *partitions*)

Bases: object

Parameters

- **name** (*str*) –
- **partitions** (*list*[[hallyd.fs.Path](#)]) –

create(*hostname*, *, *do_create*=*True*)

Parameters

- **hostname** (*str*) –
- **do_create** (*bool*) –

Return type[RaidPartition](#)

class [hallyd.disk._PartitionSetup](#)(*, *fs_type*=*None*, *mountpoint*=*None*, *label*=*None*, *do_format*=*True*)

Bases: [object](#)

Base class for all kinds of partitions.

Parameters

- **fs_type** (*Optional*[[_PartitionType](#)]) – The filesystem type name.
- **mountpoint** (*Optional*[*str*]) – The mountpoint to use .
- **label** (*Optional*[*str*]) – The partition label to assign.
- **do_format** (*bool*) – If to format the partition (or leaving it with its existing data).

make_filesystem(*partition_dev_path*)

Parameters

partition_dev_path (*Union*[*str*, [Path](#)]) –

Return type*None*

mountpoint_spec(*partition*)

Parameters

partition ([Partition](#)) –

Return type*Optional*[[Mountpoint](#)]

class [hallyd.disk._PartitionType](#)(*gpt_uuid*, *mbr_id*=*None*, *mkfs_command*=*None*, *fstab_type_name*=*None*)

Bases: [object](#)

Parameters

- **gpt_uuid** (*str*) –
- **mbr_id** (*Optional*[*str*]) –
- **mkfs_command** (*Optional*[*list*[*str*]]) –
- **fstab_type_name** (*Optional*[*str*]) –

static **by_gpt_uuid**(*parttype_uuid*)

Parameters

parttype_uuid (*str*) –

Return type*Optional*[[_PartitionType](#)]

property `fstab_type_name`

property `gpt_uuid`

make_filesystem(*partition_path*)

Parameters

partition_path (*Union[str, Path]*) –

property `mbr_id`

`hallyd.disk._disks_sort_key`(*disk*)

Parameters

disk (*Disk*) –

`hallyd.disk._find_disk_for_setup`(*disks, disk_setup*)

Parameters

- **disks** (*list[hallyd.disk.Disk]*) –

- **disk_setup** (*DiskSetup*) –

Return type

Disk

`hallyd.disk._lsblk`(*params*)

Return type

dict[str, Any]

`hallyd.disk.combine_disks_to_setups`(*disks, disk_setups*)

Parameters

- **disks** (*list[hallyd.disk.Disk]*) –

- **disk_setups** (*list[hallyd.disk.DiskSetup]*) –

Return type

list[hallyd.disk.DiskIntent]

`hallyd.disk.effective_partition_setup_order`(*partition_setups*)

Parameters

partition_setups (*list[hallyd.disk.PartitionSetup]*) –

Return type

list[hallyd.disk.OrderedPartitionSetupsEntry]

`hallyd.disk.find_disks_for_setups`(*disks, disk_setups*)

Parameters

- **disks** (*list[hallyd.disk.Disk]*) –

- **disk_setups** (*list[hallyd.disk.DiskSetup]*) –

Return type

dict[hallyd.disk.DiskSetup, hallyd.disk.Disk]

`hallyd.disk.find_partition_for_setup(disk, disk_setup, partition)`

Parameters

- **disk** ([Disk](#)) –
- **disk_setup** ([DiskSetup](#)) –
- **partition** ([PartitionSetup](#)) –

Return type

[DiskPartition](#)

`hallyd.disk.host_disks()`

Return type

list[[hallyd.disk.Disk](#)]

`hallyd.disk.host_partition_for_fs_path(fs_path)`

Parameters

fs_path ([Union\[str, Path\]](#)) –

Return type

[Partition](#)

`hallyd.disk.host_raid_partitions()`

Return type

list[[hallyd.disk.RaidPartition](#)]

`hallyd.disk.mount(dev_path, target_path)`

Parameters

- **dev_path** ([Path](#)) –
- **target_path** ([Path](#)) –

`hallyd.disk.partition_path(diskpath, part_no)`

Returns a partition device path for a given disk device path and a partition number. Examples:

- “/dev/sda”, 1 -> “/dev/sda1”
- “/dev/loop2”, 1 -> “/dev/loop2p1”

Parameters

- **diskpath** ([Path](#)) –
- **part_no** ([int](#)) –

Return type

[Path](#)

`hallyd.disk.partition_tuple(partition_dev)`

Returns the disk device path partition number for a given partition device path. Examples:

- “/dev/sda1” -> “/dev/sda”, 1
- “/dev/loop2p1” -> “/dev/loop2”, 1

Parameters

partition_dev ([Union\[str, Path\]](#)) –

Return type

tuple[*hallyd.fs.Path*, int]

`hallyd.disk.raid_setups_from_disk_intents(disk_intents)`

Parameters

disk_intents (list[*hallyd.disk.DiskIntent*]) –

Return type

dict[str, *hallyd.disk.RaidSetup*]

`hallyd.disk.reload_devices()`

`hallyd.disk.umount(path)`

Parameters

path (*Path*) –

4.1.7 hallyd.fs module

class `hallyd.fs.OnRemoveError`(*value*, *names=None*, *, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

FAIL_INSTANTLY = 3

SKIP_AND_FAIL_LATER = 2

SKIP_AND_IGNORE = 1

class `hallyd.fs.OnRemovePassingFileSystemBoundary`(*value*, *names=None*, *, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

CONTINUE_REMOVING_BEHIND_BOUNDARY = 2

ERROR = 1

TRY_UNMOUNTING = 3

TRY_UNMOUNTING_FORCEFULLY = 4

class `hallyd.fs.Path`(*args, **kwargs)

Bases: *PosixPath*

Construct a *PurePath* from one or several strings and or existing *PurePath* objects. The strings and path objects are combined so as to yield a canonicalized path, which is incorporated into the new *PurePath* object.

__change_access__call_ch_func(*args, **kwargs)

__copy(*destination*, *, *remove_source*, *sparse*, *transfer_perms*, *mode*, *owner*, *group*, *readable_by_all*, *executable*)

Parameters

- **source** (*Path*) –
- **destination** (*Path*) –

- **remove_source** (*bool*) –
- **sparse** (*bool*) –
- **transfer_perms** (*bool*) –
- **mode** (*Optional[Union[str, int, bool]]*) –
- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

Return type

None

__fallback_safely_create(**, mode=416, owner=True, group=True, readable_by_all=None, executable=None*)

Parameters

- **destination** (*Path*) –
- **mode** (*Optional[Union[str, int, bool]]*) –
- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

__file_ends_with_newline()

Parameters

fd (*int*) –

Return type

bool

__open(**, exist_ok, not_exist_ok, create_directory*)

Parameters

- **exist_ok** (*bool*) –
- **not_exist_ok** (*bool*) –
- **create_directory** (*bool*) –

Return type

tuple[int, bool, bool]

__parse_ownership(*lookup_func, true_result, none_result*)

Parameters

- **owner** (*Optional[Union[str, int, bool]]*) –
- **lookup_func** (*Callable*) –
- **true_result** (*int*) –
- **none_result** (*int*) –

Return type

int

`__parse_permission_mode(was_created, fd_stat, readable_by_all, executable)`**Parameters**

- `mode` (*Optional*[*Union*[*str*, *int*, *bool*]]) –
- `was_created` (*bool*) –
- `fd_stat` (*stat_result*) –
- `readable_by_all` (*Optional*[*bool*]) –
- `executable` (*Optional*[*bool*]) –

Return type

int

`__parse_permission_mode__part_re = re.compile('([ugoa]*)([+--]?)([ugorwxXst]*)')``__parse_permission_mode__rmask_for_letters = {'r': 292, 's': 3072, 't': 512, 'w': 146, 'x': 73}``__parse_permission_mode__str_helper(current_mode, is_directory)`**Parameters**

- `mode_str` (*str*) –
- `current_mode` (*int*) –
- `is_directory` (*bool*) –

Return type

int

`__parse_permission_mode__str_helper__expand(lvalue)`**Parameters**

- `mode` (*int*) –
- `lvalue` (*str*) –

Return type

int

`__parse_permission_mode__str_helper__lmask()`**Parameters**`lvalue` (*str*) –**Return type**

int

`__parse_permission_mode__str_helper__rmask(current_mode, is_directory)`**Parameters**

- `rvalue` (*str*) –
- `current_mode` (*int*) –
- `is_directory` (*bool*) –

Return type

int

__path_depth()**Parameters****path** (*Path*) –**Return type**

int

__path_trimmed_to_depth(*max_depth*)**Parameters**

- **path** (*Path*) –
- **max_depth** (*int*) –

Return type*Path***__remove__on_error_func()****Parameters****on_error** (*Union[Callable, OnRemoveError]*) –**Return type***Callable***__remove__on_passing_filesystem_boundary_func()****Parameters**

on_passing_filesystem_boundary (*Union[Callable, OnRemovePassingFileSystemBoundary]*) –

Return type*Callable***__remove__open**(*not_exist_ok, on_error, had_errors*)**__remove__subtree**(*dir_dev_id, dir_path, on_passing_filesystem_boundary, on_error, had_errors*)

__set_data(*data, *, as_directory=False, exist_ok=True, not_exist_ok=True, append=False, ensure_head_newline=True, ensure_tail_newline=True, preserve_perms=False, mode=416, owner=True, group=True, readable_by_all=None, executable=None*)

Parameters

- **data** (*Optional[Union[AnyStr, RawIOBase]]*) –
- **as_directory** (*bool*) –
- **exist_ok** (*bool*) –
- **not_exist_ok** (*bool*) –
- **append** (*bool*) –
- **ensure_head_newline** (*bool*) –
- **ensure_tail_newline** (*bool*) –
- **preserve_perms** (*bool*) –
- **mode** (*Optional[Union[str, int, bool]]*) –

- **owner** (*Optional[Union[str, int, bool]]*) –
- **group** (*Optional[Union[str, int, bool]]*) –
- **readable_by_all** (*Optional[bool]*) –
- **executable** (*Optional[bool]*) –

Return type

None

append_data(*data, *, exist_ok=True, ensure_head_newline=True, ensure_tail_newline=True, preserve_perms=False, mode=416, owner=True, group=True, readable_by_all=None, executable=None*)

Appends data to the end of the file at this path and apply some permission settings.

Per default, those permission settings will be applied even if the file already exists (see *preserve_perms*)!

Parameters

- **data** (*Union[AnyStr, RawIOBase]*) – The data to write.
- **exist_ok** (*bool*) – Whether it is okay if this file already exists.
- **ensure_head_newline** (*bool*) – For appending, whether to ensure a newline where *data* begins.
- **ensure_tail_newline** (*bool*) – For appending, whether to ensure a newline where *data* end.
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type

Self

apply_substitutions(**substitutions, source=None, preserve_perms=True, mode=416, owner=True, group=True, readable_by_all=None, executable=None*)

Apply regexp substitution patterns to this file.

Parameters

- **substitutions** (*tuple[str, str]*) – Each substitution is a tuple of the match pattern and the replacement pattern.
- **source** (*Union[str, Path]*) – The source path to read the original content from. Default: This file.
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user’s primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type[Path](#)

change_access(*mode=None, *, follow_symlinks=True, recursive=False, owner=None, group=None, readable_by_all=None, executable=None*)

Change file permissions and ownership.

Parameters

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. If *None*, leave it unchanged (unless some other flags are set).
- **follow_symlinks** (*bool*) – If this is a symlink, whether to change the permission settings of the target item instead.
- **recursive** (*bool*) – Whether to apply the specified changes to the entire subtree (useful for directories).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. If *True*, it’s the (effective) current user. If *False*, leave it unchanged. Otherwise either a UID or user name.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. If *True*, it’s the (effective) current user’s primary group. If *False*, leave it unchanged. Otherwise either a GID or group name.
- **readable_by_all** (*Optional[bool]*) – If set, the mode will be extended to readable for all (and maybe executable).
- **executable** (*Optional[bool]*) – Whether to set normal files to executable for everyone with read privileges. If *True* or *False*, enable or disable this flag. If *None*, leave untouched.

Return type[Self](#)

copy_to(*destination, *, exist_ok=False, merge=False, sparse=False, transfer_perms=False, mode=True, owner=True, group=True, readable_by_all=None, executable=None*)

Copy the item at this path to a destination.

If you copy a directory, all settings (incl. permission flags, ...) are applied to the entire tree.

Parameters

- **destination** (*Union[str, Path]*) – The destination path.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (per default it will be removed then).
- **merge** (*bool*) – Whether to merge the source directory file-wise into the destination directory, instead of relying on an empty destination.
- **sparse** (*bool*) – If to copy in sparse mode. Usually not needed.

- **transfer_perms** (*bool*) – Whether to transfer the permission settings of the source file.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#). Default: u=rw,g=r and also +x if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user’s primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type[Path](#)

expand_archive_to(*destination*, *, *take_1st_level=False*, *exist_ok=False*, *mode=True*, *owner=True*, *group=True*, *readable_by_all=None*, *executable=None*)

Expand this archive file (zip or tar-based) to a destination.

Parameters

- **destination** (*Union[str, Path]*) – The destination path.
- **take_1st_level** (*bool*) – Whether to take the top-level item from the archive (must be the only one on top level) and extract this one to the destination.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#). Default: u=rw,g=r and also +x if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user’s primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type[Path](#)

classmethod home_dir(*user=None*)

Parameters

user (*Optional[str]*) –

Return type[Path](#)

iterdir()

Iterate over the files/subdirectories/etc in this directory.

Does not include the special items . and ...

Return type[Iterable\[Path\]](#)


```
make_dir(*, until=None, exist_ok=False, parent_exist_ok=True, preserve_perms=False, mode=488,
          owner=True, group=True, readable_by_all=None)
```

Create a directory at this path.

With `readable_by_all=True` and `preserve_perms=True` this function behaves like `mkdir()`.

Parameters

- **until** (*Optional[Union[str, Path]]*) – Similar to *parents*, but only creates all super-directories up to (excluding) that one.
- **exist_ok** (*bool*) – Whether it is okay if this directory already exists (it will set permission related attributes anyway).
- **parent_exist_ok** (*bool*) – Whether it is okay if a *parent* directory up to *until* already exists (it will NOT set permission related attributes for them).
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the directory already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type

[Path](#)

```
make_file(*, exist_ok=True, preserve_perms=False, mode=416, owner=True, group=True,
          readable_by_all=None, executable=None)
```

Create a file at this path.

With `readable_by_all=True` and `preserve_perms=True` this function behaves like `touch()`.

Parameters

- **exist_ok** (*bool*) – Whether it is okay if this file already exists (it will set permission related attributes anyway).
- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type

[Path](#)

move_to(*destination*, *, *exist_ok=False*)

Move the item at this path to a destination. Permission settings will be retained.

Parameters

- **destination** (*Union[str, Path]*) – The destination path.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).

Return type

Path

TODO merge ?!

non_existent()

Return type

bool

relative_to(*other*, *strict=True*)

Return the relative path to another path identified by the passed arguments. If the operation is not possible (because this is not a subpath of the other path), raise *ValueError*.

Parameters

- **other** (*Union[str, Path]*) –
- **strict** (*bool*) –

Return type

Path

remove(*, *not_exist_ok=False*, *on_error=OnRemoveError.SKIP_AND_FAIL_LATER*,
on_passing_filesystem_boundary=OnRemovePassingFileSystemBoundary.TRY_UNMOUNTING)

Remove the item at this path (for directories, including the entire tree).

Parameters

- **not_exist_ok** (*bool*) – Whether it is okay if this item does not exist.
- **on_error** (*Union[Callable, OnRemoveError]*) – How to behave when errors occur.
- **on_passing_filesystem_boundary** (*Union[Callable, OnRemovePassingFileSystemBoundary]*) – How to behave when the removal of a directory tree would pass a filesystem boundary (i.e. if there is some other filesystem mounted somewhere in the tree).

Return type

Self

set_data(*data*, *, *exist_ok=True*, *preserve_perms=False*, *mode=416*, *owner=True*, *group=True*,
readable_by_all=None, *executable=None*)

Write data into a file at this path and apply some permission settings.

Per default, those permission settings will be applied even if the file already exists (see *preserve_perms*)!

With *readable_by_all=True* and *preserve_perms=True* this function behaves like *write_text()* or *write_bytes()*.

Parameters

- **data** (*Union[AnyStr, RawIOBase]*) – The data to write.
- **exist_ok** (*bool*) – Whether it is okay if this file already exists.

- **preserve_perms** (*bool*) – Whether to leave permission settings untouched if the file already exists.
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **executable** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.

Return type*Self*

classmethod temp_dir(**, mode=488, owner=True, group=True, readable_by_all=None, temp_root_path='/tmp'*)

Create a fresh temporary directory and return its path. You must use it for a with-block; it will be removed after that block in usual cases. Otherwise at a somewhat later time (usually after the process terminated). This removal will fail if you do not have permissions to do so.

Parameters

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See [change_access\(\)](#).
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See [change_access\(\)](#). Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See [change_access\(\)](#). Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See [change_access\(\)](#). Default has no effect.
- **temp_root_path** (*Union[str, Path]*) – The parent directory where to create the temporary directory.

Return type*ContextManager[Self]*

class hallyd.fs._Archive(*archive_fileobj*)

Bases: ABC

Parameters

archive_fileobj (*RawIOBase*) –

_abc_impl = *<_abc._abc_data object>*

abstract classmethod check_file_supported_by_begin(*begin*)

Parameters

begin (*bytes*) –

Return type*bool*

abstract extract_all(*destination*)

Parameters

destination (*Path*) –

Return type

None

abstract property take_1st_level_default: bool**class** hallyd.fs._DiskSpaceInfo(*total: int, used: int, free: int*)

Bases: object

Parameters

- **total** (*int*) –
- **used** (*int*) –
- **free** (*int*) –

free: int**total: int****used: int****class** hallyd.fs._TarArchive(*archive_fileobj*)Bases: [_Archive](#)**Parameters****archive_fileobj** (*RawIOBase*) –**_abc_impl** = <_abc._abc_data object>**classmethod** **check_file_supported_by_begin**(*begin*)**extract_all**(*destination*)**take_1st_level_default** = True**class** hallyd.fs._ZipArchive(*archive_fileobj*)Bases: [_Archive](#)**Parameters****archive_fileobj** (*RawIOBase*) –**_abc_impl** = <_abc._abc_data object>**classmethod** **check_file_supported_by_begin**(*begin*)**extract_all**(*destination*)**take_1st_level_default** = False**hallyd.fs._archive**(*archive_fileobj*)**hallyd.fs.byte_size_to_human_readable**(*size*)

Return a human readable format for a size in bytes.

Parameters**size** (*int*) – The number to format in bytes.**Return type**

str

`hallyd.fs.disk_space(fs_root_dir)`

Parameters

fs_root_dir (*Union[str, Path]*) –

`hallyd.fs.disk_usage(path)`

Return the disk usage for a file or directory.

Parameters

path (*Union[str, Path]*) – The path.

Return type

int

`hallyd.fs.expand_archive(source, destination, *, take_1st_level=None, exist_ok=False, mode=True, owner=True, group=True, readable_by_all=None, executable=None)`

Expand an archive file (zip or tar-based) to a destination.

Parameters

- **source** (*Union[str, Path, RawIOBase]*) – The source archive to expand.
- **destination** (*Union[str, Path]*) – The destination path.
- **take_1st_level** (*Optional[bool]*) – Whether to take the top-level item from the archive (must be the only one on top level) and extract this one to the destination. If None: Depends on the archive format.
- **exist_ok** (*bool*) – Whether it is not an error if the destination path already exists (it will be removed then).
- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See `change_access()`. Default: `u=rw,g=r` and also `+x` if the source is executable.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See `change_access()`. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See `change_access()`. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See `change_access()`. Default has no effect.
- **executable** (*Optional[bool]*) – See `change_access()`. Default has no effect.

Return type

[Path](#)

`hallyd.fs.temp_dir(*, mode=488, owner=True, group=True, readable_by_all=None, temp_root_path='/tmp')`

Create a fresh temporary directory and return its path. You must use it for a with-block; it will be removed after that block in usual cases. Otherwise, at a somewhat later time (usually after the process terminated). This removal will fail if you do not have permissions to do so.

Parameters

- **mode** (*Optional[Union[str, int, bool]]*) – The permission mode to set. See `change_access()`.
- **owner** (*Optional[Union[str, int, bool]]*) – The item owner. See `change_access()`. Default: Current (effective) user.
- **group** (*Optional[Union[str, int, bool]]*) – The item group. See `change_access()`. Default: Current (effective) user's primary group.
- **readable_by_all** (*Optional[bool]*) – See `change_access()`. Default has no effect.

- **temp_root_path** (*Union[str, Path]*) – The parent directory where to create the temporary directory.

Return type*ContextManager*[*Path*]

4.1.8 hallyd.fs_monitor module

class hallyd.fs_monitor.**FilesystemMonitor**(*paths, trigger_initially=True)

Bases: ABC

Parameters

- **paths** (*_fs.TInputPath*) –
- **trigger_initially** (*bool*) –

class **_Thread**(paths, trigger_initially, monitor)

Bases: Thread

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

- **paths** (*Iterable[_fs.Path]*) –
- **trigger_initially** (*bool*) –
- **monitor** (*FilesystemMonitor*) –

check_if_changed(wait_until_handled)**Parameters****wait_until_handled** (*bool*) –**Return type**

None

force_changed(wait_until_handled)**Parameters****wait_until_handled** (*bool*) –**Return type**

None

run()

Method representing the thread’s activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object’s constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

```

    stop()

    _abc_impl = <_abc._abc_data object>

    abstract _changed()

        Return type
        None

    check_if_changed(*, wait_until_handled=True)

        Parameters
        wait_until_handled (bool) –

        Return type
        None

    force_changed(*, wait_until_handled=True)

        Parameters
        wait_until_handled (bool) –

        Return type
        None

    property paths: Iterable[Path]

class hallyd.fs_monitor.Watcher
    Bases: ABC
    _abc_impl = <_abc._abc_data object>
    abstract wait_changed(*, timeout=None)

        Parameters
        timeout (Optional[float]) –

        Return type
        bool

class hallyd.fs_monitor._Watcher(paths, trigger_initially)
    Bases: Watcher
    Parameters
    • paths (Iterable[_fs.TInputPath]) –
    • trigger_initially (bool) –

class PollingWatchThread(watcher)
    Bases: WatchThread

    This constructor should always be called with keyword arguments. Arguments are:

    group should be None; reserved for future extension when a ThreadGroup class is implemented.

    target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

    name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a
    small decimal number.

    args is a list or tuple of arguments for the target invocation. Defaults to ().

    kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

```

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

watcher ([_Watcher](#)) –

_abc_impl = <_abc._abc_data object>

_wait_event()

class WatchThread(watcher)

Bases: Thread, ABC

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

watcher ([_Watcher](#)) –

_abc_impl = <_abc._abc_data object>

_setup()

_teardown()

_wait_event()

Return type

bool

property _watcher: [_Watcher](#)

run()

Method representing the thread’s activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object’s constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()

classmethod try_create(watcher)

Parameters

watcher ([_Watcher](#)) –

Return type

Optional[[WatchThread](#)]

class WatchdogWatchThread(watcher)

Bases: [WatchThread](#)

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

watcher ([_Watcher](#)) –

_abc_impl = <_abc._abc_data object>

_setup()

_teardown()

_WatchThreadTypes = (<class 'hallyd.fs_monitor._Watcher.PollingWatchThread'>, <class 'hallyd.fs_monitor._Watcher.WatchdogWatchThread'>)

_abc_impl = <_abc._abc_data object>

_check_if_changed(*, wait_until_handled=False)

Parameters

wait_until_handled (bool) –

Return type

None

_force_changed(*, wait_until_handled=False)

Parameters

wait_until_handled (bool) –

Return type

None

property paths: [Iterable](#)[[Path](#)]

wait_changed(*, timeout=None)

hallyd.fs_monitor.watch(*paths, trigger_initially=True)

Parameters

• **paths** ([Union](#)[[str](#), [Path](#)]) –

• **trigger_initially** (bool) –

Return type

[Watcher](#)

4.1.9 hallyd.io module

```
class hallyd.io.Lock
    Bases: ABC
    _abc_impl = <_abc._abc_data object>
    abstract acquire(blocking=True, timeout=-1)

        Parameters
            • blocking (bool) –
            • timeout (float) –

        Return type
            bool
    abstract locked()

        Return type
            bool
    abstract release()

        Return type
            None

class hallyd.io._Lock(lock_path, is_reentrant, peek_interval)
    Bases: Lock
    Parameters
        • lock_path (_fs.Path) –
        • is_reentrant (bool) –
        • peek_interval (float) –
    _abc_impl = <_abc._abc_data object>
    acquire(blocking=True, timeout=-1)
    locked()
    release()

class hallyd.io._LoopDevice(**kwargs)
    Bases: object
    property back_file: Path
    detach()
    property dev_path: Path

hallyd.io._detach_loop_device(dev_path, back_file)
hallyd.io.all_loop_devices()
```

`hallyd.io.connect_diskimage(disk_image_path)`

Parameters

`disk_image_path` (`Path`) –

Return type

`ContextManager[Path]`

`hallyd.io.connect_diskimage_buffered(dev_path, *, buffer_size_gb)`

Parameters

- `dev_path` (`Path`) –
- `buffer_size_gb` (`float`) –

Return type

`ContextManager[Path]`

`hallyd.io.create_diskimage(path, *, size_gb)`

Parameters

- `path` (`Union[str, Path]`) –
- `size_gb` (`float`) –

Return type

`None`

`hallyd.io.lock(lock_path, *, is_reentrant=True, peek_interval=0.25)`

Parameters

- `lock_path` (`Union[str, Path]`) –
- `is_reentrant` (`bool`) –
- `peek_interval` (`float`) –

Return type

`Lock`

`hallyd.io.loop_device_by_dev_path(dev_path)`

Parameters

`dev_path` (`Path`) –

Return type

`Optional[_LoopDevice]`

4.1.10 hallyd.ipc module

exception `hallyd.ipc.BadConnectionError(message)`

Bases: `OSError`

Parameters

`message` (`str`) –

class `hallyd.ipc.Enablabile`

Bases: `ABC`

```
_abc_impl = <_abc._abc_data object>

abstract disable()

    Return type
    None

abstract enable()

    Return type
    None

abstract property is_enabled: bool

exception hallyd.ipc.IPCServerPathAlreadyExistsError(ipc_path)
    Bases: BadConnectionError

    Parameters
    ipc_path (Path) –

    property ipc_path

exception hallyd.ipc.IPCServerUnavailableError(ipc_path)
    Bases: BadConnectionError

    Parameters
    ipc_path (Path) –

    property ipc_path

exception hallyd.ipc.MethodCallErroneousError(message)
    Bases: RuntimeError

    Parameters
    message (str) –

class hallyd.ipc._Client
    Bases: object

    class Proxy(request_func)
        Bases: object

        abstract _request(method_name, args, kwargs)

        property object

class hallyd.ipc._LocalClient(path)
    Bases: _Client

    Parameters
    path (Path) –

    _request(method_name, args, kwargs)

class hallyd.ipc._NetworkClient(connection, path)
    Bases: _Client

    Parameters
    • connection (_net.Connection) –
    • path (Path) –
```

_request(*method_name*, *args*, *kwargs*)

class hallyd.ipc._ThreadedServer(*object*, *, *path*)

Bases: [Enablable](#)

Parameters

- **object** (*Any*) –
- **path** ([Path](#)) –

class _MainThread(*server*)

Bases: Thread

This constructor should always be called with keyword arguments. Arguments are:

group should be None; reserved for future extension when a ThreadGroup class is implemented.

target is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to ().

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__()) before doing anything else to the thread.

Parameters

server ([_ThreadedServer](#)) –

run()

Method representing the thread’s activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object’s constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()

class _Request(*connection*, *method_name*, *args*, *kwargs*)

Bases: object

Parameters

- **method_name** (*str*) –
- **args** (*list[Optional[Any]]*) –
- **kwargs** (*dict[str, list[Optional[Any]]]*) –

answer(*, *result=None*, *error=None*)

_ThreadedServer__get_request(*connection*)

Return type

Optional[[_Request](#)]

_ThreadedServer__worker(*request*, *connection*)

_abc_impl = <_abc._abc_data object>

`_try_process_request()`

Return type

None

`disable()`

`enable()`

property `is_enabled`

`hallyd.ipc._socket_path(ipc_path)`

Parameters

`ipc_path` ([Path](#)) –

Return type

[Path](#)

`hallyd.ipc.client(path, *, connection=None)`

Parameters

- `path` ([Path](#)) –
- `connection` (*Optional* [[Connection](#)]) –

Return type

Any

`hallyd.ipc.threaded_server(object_, *, path)`

Parameters

- `object_` (*Any*) –
- `path` ([Path](#)) –

Return type

[Enablable](#)

4.1.11 hallyd.ipc_hub module

`class hallyd.ipc_hub.Hub(path)`

Bases: `Generic[_RequestT, _ResponseT]`

Hubs provide one way of distributed computing via `hallyd.ipc`.

Clients of a hub can put requests on a hub, and eventually get answer data.

A hub contains an IPC server for an internal hub controller, which provides a pluggable interface. Connect [HubWorker](#) instances to the hub for request processing.

You must use it for a with-block.

Parameters

`path` ([Path](#)) –

`__verify_entered()`

get_answer(*request_id*)

Return the answer for a request made in the past.

This can only be called for each request.

Parameters

request_id (*int*) – The request id.

Return type

_ResponseT

property path

The IPC path where hub workers can be plugged into by means of *HubWorker.plugin_into_hub()*.

put_request(*request*)

Make a request to the hub and return the request id for further actions.

Parameters

request (*_RequestT*) –

Return type

int

class hallyd.ipc_hub.**HubWorker**(*ipc_dialog_hub_object*, *, *poll_interval=0.5*)

Bases: *Generic[_RequestT, _ResponseT]*

Hub workers can be plugged into *Hub* instances in order to implement a routine that processes requests.

Plug a worker into a hub using *plugin_into_hub()*.

Parameters

- **ipc_dialog_hub_object** (*_HubIpcObject*) –
- **poll_interval** (*float*) –

__run_worker_thread()

answer_request(*request_id*, *response*)

Parameters

- **request_id** (*int*) –
- **response** (*_ResponseT*) –

Return type

bool

classmethod **plugin_into_hub**(*path*, *connection=None*, ***kwargs*)

Creates a new instance of this hub worker type and connects it to a hub while active.

You must use it for a with-block.

Parameters

- **path** (*_fs.Path*) –
- **connection** (*Optional[_net.Connection]*) –
- **kwargs** –

Return type

t.Self

abstract request_arrived(*request*)

Parameters

request (*_RequestT*) –

Return type

None

request_disappeared(*request*)

Parameters

request (*_RequestT*) –

Return type

None

try_lock_request(*request*)

Parameters

request (*_RequestT*) –

Return type

bool

class hallyd.ipc_hub._HubIpObject

Bases: Generic[_RequestT, _ResponseT]

The internal controller used by each hubs in order to provide pluggability.

add_answer(*request_id*, *response*)

Store the answer for a request.

Parameters

- **request_id** (*int*) –
- **response** (*_ResponseT*) –

Return type

bool

add_request(*request*)

Add a new request to the hub.

Parameters

request (*_RequestT*) –

Return type

int

pending_request_ids()

Return request ids for all requests that a currently open.

pop_answer(*request_id*)

Return and delete the answer for a request.

Parameters

request_id (*int*) –

Return type

_ResponseT

request_by_id(*request_id*)

Return the request by request id.

Parameters

request_id (*int*) –

Return type

Optional[*_HubRequest*[*_RequestT*]]

class hallyd.ipc_hub._HubRequest(*request_id, payload*)

Bases: *Generic*[*_RequestT*]

One request that was made to the hub.

property id

property payload

property request_id

property request_payload

4.1.12 hallyd.lang module

class hallyd.lang.AllAbstractMethodsProvidedByTrick

Bases: *Generic*[*_T*]

class hallyd.lang.Counter

Bases: *object*

next()

class hallyd.lang._AllAbstractMethodsProvidedByTrickMeta(*name, bases, namespace*)

Bases: *ABCMeta*, *Generic*[*_T*]

class hallyd.lang._ExecuteParallelThread(*fct*)

Bases: *Thread*

This constructor should always be called with keyword arguments. Arguments are:

group should be *None*; reserved for future extension when a *ThreadGroup* class is implemented.

target is the callable object to be invoked by the *run()* method. Defaults to *None*, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to *()*.

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to *{}*.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (*Thread.__init__()*) before doing anything else to the thread.

Parameters

fct (*Callable*[[*[], None*]] –

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard `run()` method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the `args` and `kwargs` arguments, respectively.

```
hallyd.lang.call_now_with_retry(*, tries=8, interval=30, interval_fact=1, retry_on=None)
```

Parameters

- **tries** (*int*) –
- **interval** (*float*) –
- **interval_fact** (*float*) –
- **retry_on** (*Optional[Iterable[type[Exception]]]*) –

Return type

Callable

```
hallyd.lang.execute_in_parallel(funcs)
```

Parameters

funcs (*list[Callable[[], NoneType]]*) –

Return type

None

```
hallyd.lang.unique_id(*, numeric_only=False)
```

Parameters

numeric_only (*bool*) –

Return type

str

```
hallyd.lang.with_friendly_repr_implementation(*, skip=())
```

Parameters

skip (*Iterable[str]*) –

```
hallyd.lang.with_retry(*, tries=8, interval=30, interval_fact=1, retry_on=None)
```

Parameters

- **tries** (*int*) –
- **interval** (*float*) –
- **interval_fact** (*float*) –
- **retry_on** (*Optional[Iterable[type[Exception]]]*) –

Return type

Callable

4.1.13 hallyd.net module

exception `hallyd.net.AccessDeniedError`

Bases: `OSError`

class `hallyd.net.Connection`

Bases: `object`

class `ExecutionResult`(*returncode, output, error_output*)

Bases: `object`

Parameters

- `returncode` (*int*) –
- `output` (*str*) –
- `error_output` (*str*) –

property `error_output`: `str`

property `output`: `str`

property `returncode`

abstract `exec`(*command*)

Parameters

`command` (*list[str]*) –

Return type

`ExecutionResult`

is_alive()

Return type

`bool`

abstract `mount`(*remote_path, local_path*)

Parameters

- `remote_path` (`Path`) –
- `local_path` (`Path`) –

Return type

`None`

abstract `umount`(*local_path*)

Parameters

`local_path` (`Path`) –

Return type

`None`

exception `hallyd.net.CouldNotConnectError`

Bases: `OSError`

```
class hallyd.net.SshConnection(host, *, port, user, password, connect_timeout=60)
```

Bases: [Connection](#)

Parameters

- **host** (*str*) –
- **port** (*int*) –
- **user** (*str*) –
- **password** (*str*) –
- **connect_timeout** (*float*) –

exec(*command*)

mount(*remote_path*, *local_path*)

umount(*local_path*)

4.1.14 hallyd.py_import module

```
hallyd.py_import._file_name_to_module_name(file_name)
```

Parameters

file_name (*str*) –

Return type

str

```
hallyd.py_import.import_module(module_file, *, module_name_left='+hallyd..', module_name_right=None,  
                               using_sys_modules=False)
```

Parameters

- **module_file** (*Union[str, Path]*) –
- **module_name_left** (*str*) –
- **module_name_right** (*Optional[str]*) –
- **using_sys_modules** (*bool*) –

Return type

module

```
hallyd.py_import.import_types_from_module_dir(base_type, module_dir, *,  
                                              file_name_glob_pattern='*.[Pp][Yy]',  
                                              module_name_left='+hallyd..',  
                                              using_sys_modules=False)
```

Parameters

- **base_type** (*type[~T]*) –
- **module_dir** (*Union[str, Path]*) –
- **file_name_glob_pattern** (*str*) –
- **module_name_left** (*str*) –
- **using_sys_modules** (*bool*) –

Return type*Iterable[type[~T]]*`hallyd.py_import.types_from_module(base_type, module)`**Parameters**

- **base_type** (*type[~T]*) –
- **module** (*module*) –

Return type*Iterable[type[~T]]*

4.1.15 hallyd.services module

`class hallyd.services.CalendarTask`Bases: `_Task`, `ABC``_abc_impl = <_abc._abc_data object>`

```
class hallyd.services.CalendarTaskSetup(_name: Optional[str] = None, _runnable:
Optional[ForwardRef('TRunnable')] = None, _description:
Optional[str] = None, _user: Union[str, int, NoneType] = None,
_group: Union[str, int, NoneType] = None, _discard_output:
bool = False, _working_dir: hallyd.fs.Path = Path('/'),
_calendars: Iterable[ForwardRef('_Calendar')] = ())
```

Bases: `_TimedTaskSetup`**Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[TRunnable]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (*Path*) –
- **_calendars** (*Iterable[_Calendar]*) –

`class _Calendar`Bases: `object``class _DailyCalendar(time: Optional[str])`Bases: `_Calendar`**Parameters****time** (*Optional[str]*) –**time:** `Optional[str]`

```
class _MonthlyCalendar(time: Optional[str], day: int)
    Bases: _Calendar
        Parameters
            • time (Optional[str]) –
            • day (int) –
        day: int
        time: Optional[str]
class _WeeklyCalendar(time: Optional[str], day_of_week: int)
    Bases: _Calendar
        Parameters
            • time (Optional[str]) –
            • day_of_week (int) –
        day_of_week: int
        time: Optional[str]
class _YearlyCalendar(time: Optional[str], day: int, month: int)
    Bases: _Calendar
        Parameters
            • time (Optional[str]) –
            • day (int) –
            • month (int) –
        day: int
        month: int
        time: Optional[str]
_calendars: Iterable[_Calendar] = ()
schedule_daily(*, time=None)
    Parameters
        time (Optional[str]) –
    Return type
        None
schedule_monthly(*, time=None, day=1)
    Parameters
        • time (Optional[str]) –
        • day (int) –
    Return type
        None
```

schedule_weekly(* , time=None, day_of_week=7)

Parameters

- **time** (*Optional[str]*) –
- **day_of_week** (*int*) –

Return type

None

schedule_yearly(* , time=None, day=1, month=1)

Parameters

- **time** (*Optional[str]*) –
- **day** (*int*) –
- **month** (*int*) –

Return type

None

class hallyd.services.IntervalTask

Bases: [_Task](#), ABC

_abc_impl = *<_abc._abc_data object>*

class hallyd.services.IntervalTaskSetup(*_name: Optional[str] = None, _runnable: Optional[ForwardRef('TRunnable')] = None, _description: Optional[str] = None, _user: Union[str, int, NoneType] = None, _group: Union[str, int, NoneType] = None, _discard_output: bool = False, _working_dir: [hallyd.fs.Path](#) = Path(''), _intervals: Iterable[float] = ()*)

Bases: [_TimedTaskSetup](#)

Parameters

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, [Runnable](#), Callable[[], None], Callable[[[Runnable](#)], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** ([Path](#)) –
- **_intervals** (*Iterable[float]*) –

_intervals: *Iterable[float] = ()*

schedule_by_interval(*seconds=0, *, minutes=0, hours=0*)

Parameters

- **seconds** (*float*) –
- **minutes** (*float*) –

- **hours** (*float*) –

Return type

None

class hallyd.services.NextBootTaskBases: *_Task*, ABC**_abc_impl** = <_abc._abc_data object>**class** hallyd.services.NextBootTaskSetup(*_name: Optional[str] = None, _runnable:**Optional[ForwardRef('TRunnable')] = None, _description:**Optional[str] = None, _user: Union[str, int, NoneType] = None,**_group: Union[str, int, NoneType] = None, _discard_output:**bool = False, _working_dir: hallyd.fs.Path = Path('/'),**_dependencies: Iter-**able[hallyd.services._TaskSetup._WithDependencies.Dependency]**= (), _interactive: bool = False)*Bases: *_WithDependencies*, *_TaskSetup***Parameters**

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –
- **_user** (*Optional[Union[str, int]]*) –
- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (*Path*) –
- **_dependencies** (*Iterable[Dependency]*) –
- **_interactive** (*bool*) –

_after_create()**_interactive:** *bool* = False**run_interactively**(*run_interactively=True*)**Parameters****run_interactively** (*bool*) –**Return type**

None

class hallyd.services.Runnable

Bases: ABC

exception _FinishAndReboot

Bases: BaseException

_abc_impl = <_abc._abc_data object>

reboot()

Return type

None

abstract run()

Return type

None

class hallyd.services.**Service**

Bases: [_Task](#), [ABC](#)

_abc_impl = <_abc._abc_data object>

abstract is_active()

Return type

None

abstract reload()

Return type

None

abstract restart()

Return type

None

abstract start()

Return type

None

abstract stop()

Return type

None

class hallyd.services.**ServiceSetup**(*_name: Optional[str] = None, _runnable: Optional[ForwardRef('TRunnable')] = None, _description: Optional[str] = None, _user: Union[str, int, NoneType] = None, _group: Union[str, int, NoneType] = None, _discard_output: bool = False, _working_dir: hallyd.fs.Path = Path('/'), _dependencies: Iterable[hallyd.services._TaskSetup._WithDependencies.Dependency] = (), _start_instantly: bool = True, _enabled: bool = True, _restart_delay: Optional[int] = 2, _as_one_shot: bool = False, _startup_context: Optional[Iterable[str]] = None, _post_stop: Optional[ForwardRef('TCmdLineRunnable')] = None, _options: Iterable[Callable[[str], NoneType]] = ())*

Bases: [_WithDependencies](#), [_TaskSetup](#)

Parameters

- **_name** (*Optional[str]*) –
- **_runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **_description** (*Optional[str]*) –

- `_user` (*Optional*[*Union*[*str*, *int*]]) –
- `_group` (*Optional*[*Union*[*str*, *int*]]) –
- `_discard_output` (*bool*) –
- `_working_dir` (*Path*) –
- `_dependencies` (*Iterable*[*Dependency*]) –
- `_start_instantly` (*bool*) –
- `_enabled` (*bool*) –
- `_restart_delay` (*Optional*[*int*]) –
- `_as_oneShot` (*bool*) –
- `_startup_context` (*Optional*[*Iterable*[*str*]]) –
- `_post_stop` (*Optional*[*Union*[*Iterable*[*Union*[*str*, *Path*]], *str*, *Path*]]) –
- `_options` (*Iterable*[*Callable*[[*str*], *None*]]) –

`__DEFAULT_RESTART_DELAY = 2`

`_after_create()`

`_as_oneShot: bool = False`

`_enabled: bool = True`

`_options: Iterable[Callable[[str], None]] = ()`

`_post_stop: Optional[TCmdLineRunnable] = None`

`_restart_delay: Optional[int] = 2`

`_start_instantly: bool = True`

`_startup_context: Optional[Iterable[str]] = None`

`as_oneShot(as_oneShot=True)`

Parameters

`as_oneShot` (*bool*) –

Return type

 None

`do_not_restart(do_not_restart=True)`

Parameters

`do_not_restart` (*bool*) –

Return type

 None

`do_not_start_instantly(do_not_start_instantly=True, *, not_even_enable=None)`

Parameters

- `do_not_start_instantly` (*bool*) –
- `not_even_enable` (*Optional*[*bool*]) –

Return type

None

with_option(*option*)**Parameters****option** (*Callable*[[*str*], None]) –**Return type**

None

with_post_stop_command(*post_stop_command*)**Parameters****post_stop_command** (*str*) –**Return type**

None

with_restart_delay(*restart_delay*)**Parameters****restart_delay** (*Optional*[*int*]) –**Return type**

None

with_startup_context(*startup_context*)**Parameters****startup_context** (*str*) –**Return type**

None

class hallyd.services._CalendarTaskBackend

Bases: ABC

_abc_impl = <_abc._abc_data object>**abstract** **calendar_task**(*name*)**Parameters****name** (*str*) –**Return type**[CalendarTask](#)**abstract** **create_calendar_task**(*setup*)**Parameters****setup** ([CalendarTaskSetup](#)) –**Return type**

None

abstract **remove_calendar_task**(*name*)**Parameters****name** (*str*) –**Return type**

None

```
class hallyd.services._FunctionRunnable(func)
```

Bases: [Runnable](#)

Parameters

func ([CallableRunnable](#)) –

_abc_impl = <_abc._abc_data object>

run()

```
class hallyd.services._IntervalTaskBackend
```

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract create_interval_task(*setup*)

Parameters

setup ([IntervalTaskSetup](#)) –

Return type

None

abstract interval_task(*name*)

Parameters

name (*str*) –

Return type

[IntervalTask](#)

abstract remove_interval_task(*name*)

Parameters

name (*str*) –

Return type

None

```
class hallyd.services._NextBootTaskBackend
```

Bases: ABC

_abc_impl = <_abc._abc_data object>

abstract create_next_boot_task(*setup*)

Parameters

setup ([NextBootTaskSetup](#)) –

Return type

None

abstract next_boot_task(*name*)

Parameters

name (*str*) –

Return type

[NextBootTest](#)

```
abstract remove_next_boot_action(name)
```

Parameters

name (*str*) –

Return type

None

```
class hallyd.services._ServiceBackend
```

Bases: ABC

```
_abc_impl = <_abc._abc_data object>
```

```
abstract create_service(setup)
```

Parameters

setup (*ServiceSetup*) –

Return type

None

```
abstract remove_service(name)
```

Parameters

name (*str*) –

Return type

None

```
abstract service(name)
```

Parameters

name (*str*) –

Return type

Service

```
class hallyd.services._SystemdBackend
```

Bases: *_ServiceBackend*, *_CalendarTaskBackend*, *_IntervalTaskBackend*, *_NextBootTaskBackend*

```
static _SystemdBackend__create(setup)
```

Return type

None

```
_SystemdBackend__lock = <hallyd.io._Lock object>
```

```
static _SystemdBackend__remove(service)
```

Parameters

service (*_Unit*) –

Return type

None

```
static _SystemdBackend__runnable_to_spec_info(runnable, unit_spec, service_spec_path, *,
                                              interactive=False, once=False,
                                              discard_output=False)
```

Parameters

• **runnable** (*TRunnable*) –

- `unit_spec` (`_UnitSpec`) –
- `service_spec_path` (`Optional[Path]`) –
- `interactive` (`bool`) –
- `once` (`bool`) –
- `discard_output` (`bool`) –

Return type

`str`

static `_SystemdBackend__setup_to_object_spec`(`setup`, `service_spec_path=None`)

Parameters

- `setup` (`_TaskSetup`) –
- `service_spec_path` (`Optional[Path]`) –

Return type

`_UnitSpec`

_SystemdBackend__unit(`name`, *, `type`, `also_allow_types=()`)

Parameters

- `type` (`str`) –
- `also_allow_types` (`Iterable[str]`) –

Return type

`_Unit`

class `_Unit`(`name`, `unit_type`)

Bases: `Service`, `IntervalTask`, `CalendarTask`

Parameters

- `name` (`str`) –
- `unit_type` (`str`) –

_Unit__systemctl(`*args`)

Return type

`None`

`_abc_impl = <_abc._abc_data object>`

`disable()`

`enable()`

property `full_name`

`is_active()`

property `is_enabled`

property `name`

override(*wants=()*, *requires=()*, *after=()*, *before=()*, *wanted_by=()*, *required_by=()*,
reset_wants=False, *reset_requires=False*, *reset_after=False*, *reset_before=False*,
reset_wanted_by=False, *reset_required_by=False*)

Parameters

- **wants** (*list[str]*) –
- **requires** (*list[str]*) –
- **after** (*list[str]*) –
- **before** (*list[str]*) –
- **wanted_by** (*list[str]*) –
- **required_by** (*list[str]*) –
- **reset_wants** (*bool*) –
- **reset_requires** (*bool*) –
- **reset_after** (*bool*) –
- **reset_before** (*bool*) –
- **reset_wanted_by** (*bool*) –
- **reset_required_by** (*bool*) –

Return type

None

reload()

restart()

start()

stop()

property unit_type

class _UnitSpec

Bases: object

class _MultiDict

Bases: object

add_value(*key*, *value*)

Parameters

- **key** (*str*) –
- **value** (*str*) –

clear_values(*key*)

Parameters

- key** (*str*) –

set_value(*key*, *value*)

Parameters

- **key** (*str*) –
- **value** (*str*) –

property install: [*_MultiDict*](#)

property service: [*_MultiDict*](#)

property timer: [*_MultiDict*](#)

property unit: [*_MultiDict*](#)

```
_abc_impl = <_abc._abc_data object>

static _short_name(name)

    Parameters
        name (str) –

    Return type
        tuple[str, Optional[str]]

calendar_task(name)

create_calendar_task(setup)

create_interval_task(setup)

create_next_boot_task(setup)

create_service(setup)

interval_task(name)

next_boot_task(name)

remove_calendar_task(name)

remove_interval_task(name)

remove_next_boot_action(name)

remove_service(name)

service(name)

class hallyd.services._Task
    Bases: ABC
    _abc_impl = <_abc._abc_data object>

    abstract disable()

        Return type
            None

    abstract enable()

        Return type
            None

    abstract property is_enabled: bool

    abstract property name: str

class hallyd.services._TaskSetup(_name: Optional[str] = None, _runnable:
    Optional[ForwardRef('TRunnable')] = None, _description: Optional[str]
    = None, _user: Union[str, int, NoneType] = None, _group: Union[str, int,
    NoneType] = None, _discard_output: bool = False, _working_dir:
    hallyd.fs.Path = Path(''))

    Bases: object

    Parameters
```


- `_name` (*Optional[str]*) –
- `_runnable` (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- `_description` (*Optional[str]*) –
- `_user` (*Optional[Union[str, int]]*) –
- `_group` (*Optional[Union[str, int]]*) –
- `_discard_output` (*bool*) –
- `_working_dir` (*Path*) –

```

class _WithDependencies(_dependencies:
    Iterable[hallyd.services._TaskSetup._WithDependencies.Dependency] = ())
    Bases: object
        Parameters
            _dependencies (Iterable[Dependency]) –
class Dependency(name: str, afterwards: bool, success_required: bool, optional: bool)
    Bases: object
        Parameters
            • name (str) –
            • afterwards (bool) –
            • success_required (bool) –
            • optional (bool) –
        afterwards: bool

        name: str

        optional: bool

        success_required: bool

        _dependencies: Iterable[Dependency] = ()

        add_dependency(unit_name, *, afterwards=False, success_required=False, optional=False)
            Parameters
                • unit_name (str) –
                • afterwards (bool) –
                • success_required (bool) –
                • optional (bool) –

        _after_create()

            Return type
                None

        _before_create()

            Return type
                None

        _description: Optional[str] = None

        _discard_output: bool = False

        _group: Optional[Union[str, int]] = None

```

_name: `Optional[str] = None`

_runnable: `Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]] = None`

_user: `Optional[Union[str, int]] = None`

_working_dir: `Path = Path('/')`

Parameters

paths (`Union[str, Path]`) –

Return type

`Path`

description (`description`)

Parameters

description (`str`) –

Return type

`None`

discard_output (`discard_output=True`)

Parameters

discard_output (`bool`) –

Return type

`None`

run_as_user (`user, group=None`)

Parameters

- **user** (`Union[str, int]`) –
- **group** (`Optional[Union[str, int]]`) –

Return type

`None`

run_in_working_dir (`working_dir`)

Parameters

working_dir (`Union[str, Path]`) –

Return type

`None`

class `hallyd.services._TimedTaskSetup` (`_name=None, _runnable=None, _description=None, _user=None, _group=None, _discard_output=False, _working_dir=Path('/')`)

Bases: `_TaskSetup`

Parameters

- **_name** (`Optional[str]`) –
- **_runnable** (`Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]`) –
- **_description** (`Optional[str]`) –
- **_user** (`Optional[Union[str, int]]`) –

- **_group** (*Optional[Union[str, int]]*) –
- **_discard_output** (*bool*) –
- **_working_dir** (*Path*) –

_start_instantly: *bool* = **False**

start_instantly(*start_instantly=True*)

Parameters

start_instantly (*bool*) –

Return type

None

hallyd.services._create_setup_context(*name, runnable, setup_type, create_func*)

Parameters

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –
- **setup_type** (*type[~_TTaskSetup]*) –

Return type

ContextManager[_TTaskSetup]

hallyd.services.calendar_task(*runnable*)

Parameters

runnable (*Union[CalendarTask, str]*) –

Return type

CalendarTask

hallyd.services.create_calendar_task(*name, runnable*)

Parameters

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

Return type

ContextManager[CalendarTaskSetup]

hallyd.services.create_interval_task(*name, runnable*)

Parameters

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

Return type

ContextManager[IntervalTaskSetup]

`hallyd.services.create_next_boot_task(name, runnable)`

Parameters

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

Return type

ContextManager[NextBootTaskSetup]

`hallyd.services.create_service(name, runnable)`

Parameters

- **name** (*Optional[str]*) –
- **runnable** (*Optional[Union[Iterable[Union[str, Path]], str, Path, Runnable, Callable[[], None], Callable[[Runnable], None]]]*) –

Return type

ContextManager[ServiceSetup]

`hallyd.services.interval_task(runnable)`

Parameters

runnable (*Union[IntervalTask, str]*) –

Return type

IntervalTask

`hallyd.services.next_boot_task(runnable)`

Parameters

runnable (*Union[NextBootTask, str]*) –

Return type

NextBootTask

`hallyd.services.remove_calendar_task(runnable)`

Parameters

runnable (*Union[CalendarTask, str]*) –

Return type

None

`hallyd.services.remove_interval_task(runnable)`

Parameters

runnable (*Union[IntervalTask, str]*) –

Return type

None

`hallyd.services.remove_next_boot_action(runnable)`

Parameters

runnable (*Union[NextBootTask, str]*) –

Return type

None

`hallyd.services.remove_service(runnable)`

Parameters

`runnable` (`Union[Service, str]`) –

Return type

None

`hallyd.services.service(runnable)`

Parameters

`runnable` (`Union[Service, str]`) –

Return type

`Service`

4.1.16 hallyd.subprocess module

exception `hallyd.subprocess._ProcessInfoUnavailableError`

Bases: `Exception`

exception `hallyd.subprocess._ProcessNotRunningError`

Bases: `Exception`

`hallyd.subprocess.check_call_with_stdin_string(cmd, *, stdin, **kwargs)`

Parameters

- **`cmd`** (`list`) –
- **`stdin`** (`AnyStr`) –

Return type

None

`hallyd.subprocess.check_output_with_stdin_string(cmd, *, stdin, **kwargs)`

Parameters

- **`cmd`** (`list`) –
- **`stdin`** (`AnyStr`) –

Return type

bytes

`hallyd.subprocess.is_process_running(process_permanent_id)`

Parameters

`process_permanent_id` (`str`) –

Return type

`Optional[bool]`

`hallyd.subprocess.pid_for_process_permanent_id(process_permanent_id)`

Parameters

`process_permanent_id` (`str`) –

Return type

int

`hallyd.subprocess.process_permanent_id_for_pid(pid)`

Parameters

`pid` (*int*) –

Return type

str

`hallyd.subprocess.start_function_in_new_process(func, args=None, kwargs=None, *, interactive=False, capture_output=False, shell=())`

Parameters

`func` (*Union[Callable, SupportsQualifiedName]*) –

Return type

Popen

`hallyd.subprocess.verify_tool_available(tool)`

Checks if a tool is installed (by calling it). Raises an exception if not.

Parameters

`tool` (*str*) –

Return type

None

4.1.17 hallyd.terminal module

`class hallyd.terminal.AnsiEscapedTextFormatter(*, with_rgb256)`

Bases: *TextFormatter*

Parameters

`with_rgb256` (*bool*) –

`__control_for_color(color, bare_block1_begin, bare_block2_begin, rgb256_code)`

Parameters

- `bare_block1_begin` (*int*) –
- `bare_block2_begin` (*int*) –
- `rgb256_code` (*int*) –

`_abc_impl = <_abc._abc_data object>`

`control_for_bg_color(color)`

`control_for_fg_color(color)`

`control_for_reset()`

`escape(s)`

`class hallyd.terminal.Color`

Bases: *object*

`as_bare_ansi_terminal_code()`

Return type

int

as_html()

Return type

str

as_rgb256()

Return type

tuple[float, float, float]

as_rgb_normed()

Return type

tuple[float, float, float]

as_xterm256_terminal_code()

Return type

int

by_bare_ansi_terminal_code(*code*)

by_html(*html_color*)

Parameters

html_color (str) –

Return type

Color

by_rgb256(*r*, *g*, *b*)

Parameters

- **r** (float) –
- **g** (float) –
- **b** (float) –

Return type

Color

by_rgb_normed(*r*, *g*, *b*)

Parameters

- **r** (float) –
- **g** (float) –
- **b** (float) –

Return type

Color

by_xterm256_terminal_code(*code*)

class hallyd.terminal.PlainTextFormatter

Bases: *TextFormatter*

_abc_impl = <_abc._abc_data object>

control_for_bg_color(*color*)

control_for_fg_color(*color*)

control_for_reset()

escape(*s*)

class hallyd.terminal.Style

Bases: object

has_key(*key*)

Parameters

key (*str*) –

Return type

bool

value(*key*)

Return type

Any

class hallyd.terminal.Text(*krz_shl_doc_xml*)

Bases: object

Parameters

krz_shl_doc_xml (*str*) –

__smart_strip()

Parameters

s (*str*) –

Return type

str

__traverse(*node_start_func*, *node_stop_func*)

Return type

Text

property as_xml: str

static by_plain_text(*text*, *, *smart_strip=True*)

Parameters

- **text** (*str*) –
- **smart_strip** (*bool*) –

Return type

Text

static by_xml(*krz_shl_doc_xml*)

Parameters

krz_shl_doc_xml (*str*) –

Return type

Text

format(*formatter*, *, *style*=<hallyd.terminal._DefaultStyle object>)

Parameters

- **formatter** ([TextFormatter](#)) –
- **style** ([Style](#)) –

Return type

str

indent(*width*, *, *char*=' ')

Parameters

- **width** (*int*) –
- **char** (*str*) –

Return type

[Text](#)

line_wrap_at_maximum_width(*width*)

Parameters

width (*Optional[int]*) –

Return type

[Text](#)

class hallyd.terminal.TextFormatter

Bases: [ABC](#)

_abc_impl = <_abc._abc_data object>

abstract control_for_bg_color(*color*)

Parameters

color ([Color](#)) –

Return type

str

abstract control_for_fg_color(*color*)

Parameters

color ([Color](#)) –

Return type

str

abstract control_for_reset()

Return type

str

abstract escape(*s*)

Parameters

s (*str*) –

Return type

str

```
class hallyd.terminal._BareAnsiTerminalColorSpace
```

```
    Bases: _TerminalColorSpace
```

```
    _COLORS = ((0, 0, 0), (128, 0, 0), (0, 128, 0), (128, 128, 0), (0, 0, 128), (128, 0, 128), (0, 128, 128), (192, 192, 192), (128, 128, 128), (255, 0, 0), (0, 255, 0), (255, 255, 0), (0, 0, 255), (255, 0, 255), (0, 255, 255), (255, 255, 255))
```

```
    _abc_impl = <_abc._abc_data object>
```

```
    color_to_index(r, g, b)
```

```
    index_to_color(idx)
```

```
class hallyd.terminal._DefaultStyle
```

```
    Bases: Style
```

```
class hallyd.terminal._TerminalColorSpace
```

```
    Bases: ABC
```

```
    _abc_impl = <_abc._abc_data object>
```

```
    abstract color_to_index(r, g, b)
```

Parameters

- *r* (*float*) –
- *g* (*float*) –
- *b* (*float*) –

Return type

int

```
    abstract index_to_color(idx)
```

Parameters

idx (*int*) –

Return type

tuple[float, float, float]

```
class hallyd.terminal._XTerm256TerminalColorSpace
```

```
    Bases: _TerminalColorSpace
```

```
    _COLOR_INDEX_BEGIN = 16
```

```
    _COLOR_SYMBOLS_PER_CHANNEL = 6
```

```
    _COLOR_VALUE_BEGIN = 95
```

```
    _COLOR_VALUE_END = 255
```

```
    _COLOR_VALUE_STEP = 40
```

```
    _END_INDEX_BEGIN = 256
```

```
    _GRAYSCALE_INDEX_BEGIN = 232
```

```
    _GRAYSCALE_SYMBOLS_PER_CHANNEL = 24
```

```

    _GRAYSCALE_VALUE_BEGIN = 8
    _GRAYSCALE_VALUE_END = 238
    _GRAYSCALE_VALUE_STEP = 10
    _XTerm256TerminalColorSpace__color_to_index__colors(r, g, b)
    _XTerm256TerminalColorSpace__color_to_index__grayscale(r, g, b)
    _XTerm256TerminalColorSpace__color_to_index__helper__channel(val, symbol_count,
                                                                value_begin, value_step)

    _XTerm256TerminalColorSpace__cost(c1, c2)
    _abc_impl = <_abc._abc_data object>
    color_to_index(r, g, b)
    index_to_color(idx)
hallyd.terminal.is_interactive_shell()

    Return type
        bool
hallyd.terminal.is_superuser()
hallyd.terminal.terminal_width(*,fallback_to=80)

    Parameters
        fallback_to (int) –

    Return type
        int

```

4.1.18 hallyd.text module

```

hallyd.text.match_format_string(pattern, string)

    Parameters
        • pattern (str) –
        • string (str) –

    Return type
        dict[str, str]
hallyd.text.pretty_print_xml(input_xml)

    Parameters
        input_xml (str) –

    Return type
        str

```

4.1.19 hallyd.typing module

```
class hallyd.typing.SupportsQualifiedName(*args, **kwargs)
    Bases: Protocol
    _abc_impl = <_abc._abc_data object>
    _is_protocol = True
    _is_runtime_protocol = True
```

PYTHON MODULE INDEX

h

- hallyd, 9
- hallyd._aux, 9
- hallyd._aux.services_helper__call_action, 9
- hallyd._aux.subprocess_helper__call_function, 9
- hallyd.bundle, 9
- hallyd.cleanup, 11
- hallyd.coding, 12
- hallyd.disk, 14
- hallyd.fs, 22
- hallyd.fs_monitor, 34
- hallyd.io, 38
- hallyd.ipc, 39
- hallyd.ipc_hub, 42
- hallyd.lang, 45
- hallyd.net, 47
- hallyd.py_import, 48
- hallyd.services, 49
- hallyd.subprocess, 65
- hallyd.terminal, 66
- hallyd.text, 71
- hallyd.typing, 72

Symbols

<code>_AllAbstractMethodsProvidedByTrickMeta</code> (class in <code>hallyd.lang</code>), 45	<code>attribute</code>), 70
<code>_Archive</code> (class in <code>hallyd.fs</code>), 31	<code>_GRAYSCALE_VALUE_END</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 71
<code>_BareAnsiTerminalColorSpace</code> (class in <code>hal- lyd.terminal</code>), 69	<code>_GRAYSCALE_VALUE_STEP</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 71
<code>_COLORS</code> (<code>hallyd.terminal._BareAnsiTerminalColorSpace</code> attribute), 70	<code>_HubIpcObject</code> (class in <code>hallyd.ipc_hub</code>), 44
<code>_COLOR_INDEX_BEGIN</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_HubRequest</code> (class in <code>hallyd.ipc_hub</code>), 45
<code>_COLOR_SYMBOLS_PER_CHANNEL</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_IntervalTaskBackend</code> (class in <code>hallyd.services</code>), 56
<code>_COLOR_VALUE_BEGIN</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_LocalClient</code> (class in <code>hallyd.ipc</code>), 40
<code>_COLOR_VALUE_END</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_Lock</code> (class in <code>hallyd.io</code>), 38
<code>_COLOR_VALUE_STEP</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_LoopDevice</code> (class in <code>hallyd.io</code>), 38
<code>_CalendarTaskBackend</code> (class in <code>hallyd.services</code>), 55	<code>_NetworkClient</code> (class in <code>hallyd.ipc</code>), 40
<code>_CleanupTask</code> (class in <code>hallyd.cleanup</code>), 11	<code>_NextBootTaskBackend</code> (class in <code>hallyd.services</code>), 56
<code>_Client</code> (class in <code>hallyd.ipc</code>), 40	<code>_PartitionSetup</code> (class in <code>hallyd.disk</code>), 19
<code>_Client.Proxy</code> (class in <code>hallyd.ipc</code>), 40	<code>_PartitionType</code> (class in <code>hallyd.disk</code>), 19
<code>_DefaultStyle</code> (class in <code>hallyd.terminal</code>), 70	<code>_ProcessInfoUnavailableError</code> , 65
<code>_DiskSpaceInfo</code> (class in <code>hallyd.fs</code>), 32	<code>_ProcessNotRunningError</code> , 65
<code>_END_INDEX_BEGIN</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_ServiceBackend</code> (class in <code>hallyd.services</code>), 57
<code>_ExecuteParallelThread</code> (class in <code>hallyd.lang</code>), 45	<code>_SystemdBackend</code> (class in <code>hallyd.services</code>), 57
<code>_FunctionRunnable</code> (class in <code>hallyd.services</code>), 55	<code>_SystemdBackend._Unit</code> (class in <code>hallyd.services</code>), 58
<code>_GRAYSCALE_INDEX_BEGIN</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_SystemdBackend._UnitSpec</code> (class in <code>hal- lyd.services</code>), 59
<code>_GRAYSCALE_SYMBOLS_PER_CHANNEL</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_SystemdBackend._UnitSpec._MultiDict</code> (class in <code>hallyd.services</code>), 59
<code>_GRAYSCALE_VALUE_BEGIN</code> (hal- <code>lyd.terminal._XTerm256TerminalColorSpace</code> attribute), 70	<code>_SystemdBackend.__create()</code> (hal- <code>lyd.services._SystemdBackend</code> static method), 57
	<code>_SystemdBackend.__lock</code> (hal- <code>lyd.services._SystemdBackend</code> attribute), 57
	<code>_SystemdBackend.__remove()</code> (hal- <code>lyd.services._SystemdBackend</code> static method), 57
	<code>_SystemdBackend.__runnable_to_spec_info()</code> (<code>hallyd.services._SystemdBackend</code> static method), 57
	<code>_SystemdBackend.__setup_to_object_spec()</code> (hal- <code>lyd.services._SystemdBackend</code> static method), 58
	<code>_SystemdBackend.__unit()</code> (hal-

`lyd.services._SystemdBackend` (method), 58
`_TarArchive` (class in `hallyd.fs`), 32
`_Task` (class in `hallyd.services`), 60
`_TaskSetup` (class in `hallyd.services`), 60
`_TaskSetup.WithDependencies` (class in `hallyd.services`), 61
`_TaskSetup.WithDependencies.Dependency` (class in `hallyd.services`), 61
`_TerminalColorSpace` (class in `hallyd.terminal`), 70
`_ThreadedServer` (class in `hallyd.ipc`), 41
`_ThreadedServer._MainThread` (class in `hallyd.ipc`), 41
`_ThreadedServer._Request` (class in `hallyd.ipc`), 41
`_ThreadedServer__get_request()` (`hallyd.ipc._ThreadedServer` method), 41
`_ThreadedServer__worker()` (`hallyd.ipc._ThreadedServer` method), 41
`_TimedTaskSetup` (class in `hallyd.services`), 62
`_Unit__systemctl()` (`hallyd.services._SystemdBackend._Unit` method), 58
`_WatchThreadTypes` (`hallyd.fs_monitor._Watcher` attribute), 37
`_Watcher` (class in `hallyd.fs_monitor`), 35
`_Watcher.PollingWatchThread` (class in `hallyd.fs_monitor`), 35
`_Watcher.WatchThread` (class in `hallyd.fs_monitor`), 36
`_Watcher.WatchdogWatchThread` (class in `hallyd.fs_monitor`), 36
`_WithDecorationSupportHandleMixin__decoration_ast_to_code_indexes()` (`hallyd.coding.Editor._WithDecorationSupportHandleMixin` method), 13
`_XTerm256TerminalColorSpace` (class in `hallyd.terminal`), 70
`_XTerm256TerminalColorSpace__color_to_index_color_to_index()` (`hallyd.terminal._XTerm256TerminalColorSpace` method), 71
`_XTerm256TerminalColorSpace__color_to_index_grayscale_color_to_index()` (`hallyd.terminal._XTerm256TerminalColorSpace` method), 71
`_XTerm256TerminalColorSpace__color_to_index_help_color_to_index()` (`hallyd.terminal._XTerm256TerminalColorSpace` method), 71
`_XTerm256TerminalColorSpace__cost()` (`hallyd.terminal._XTerm256TerminalColorSpace` method), 71
`_ZipArchive` (class in `hallyd.fs`), 32
`__DEFAULT_RESTART_DELAY` (`hallyd.services.ServiceSetup` attribute), 54
`__change_access__call_ch_func()` (`hallyd.fs.Path` method), 22
`__control_for_color()` (`hallyd.terminal.AnsiEscapedTextFormatter` method), 66
`__copy()` (`hallyd.fs.Path` method), 22
`__fallback_safely_create()` (`hallyd.fs.Path` method), 23
`__file_ends_with_newline()` (`hallyd.fs.Path` method), 23
`__idpath()` (`hallyd.disk.Disk` method), 14
`__indentation_for_code_fragment()` (`hallyd.coding.Editor` method), 13
`__lsblk()` (`hallyd.disk.Disk` method), 14
`__open()` (`hallyd.fs.Path` method), 23
`__parse_ownership()` (`hallyd.fs.Path` method), 23
`__parse_permission_mode()` (`hallyd.fs.Path` method), 24
`__parse_permission_mode__part_re` (`hallyd.fs.Path` attribute), 24
`__parse_permission_mode__rmask_for_letters` (`hallyd.fs.Path` attribute), 24
`__parse_permission_mode__str_helper()` (`hallyd.fs.Path` method), 24
`__parse_permission_mode__str_helper__expand()` (`hallyd.fs.Path` method), 24
`__parse_permission_mode__str_helper__lmask()` (`hallyd.fs.Path` method), 24
`__parse_permission_mode__str_helper__rmask()` (`hallyd.fs.Path` method), 24
`__path_depth()` (`hallyd.fs.Path` method), 25
`__path_trimmed_to_depth()` (`hallyd.fs.Path` method), 25
`__remove__on_error_func()` (`hallyd.fs.Path` method), 25
`__remove__on_passing_filesystem_boundary_func()` (`hallyd.fs.Path` method), 25
`__remove__open()` (`hallyd.fs.Path` method), 25
`__remove__subtree()` (`hallyd.fs.Path` method), 25
`__read_partition_table()` (`hallyd.disk.DiskIntent` method), 15
`__run_worker_thread()` (`hallyd.ipc_hub.HubWorker` method), 43
`__set_data()` (`hallyd.fs.Path` method), 25
`__smart_strip()` (`hallyd.terminal.Text` method), 68
`__strip()` (`hallyd.terminal.Text` method), 68
`__udev_property()` (`hallyd.disk.Disk` method), 14
`__verify_entered()` (`hallyd.ipc_hub.Hub` method), 42
`__write_partition_table()` (`hallyd.disk.DiskIntent` method), 15
`_abc_impl` (`hallyd.coding.Editor._ClassHandle` attribute), 12
`_abc_impl` (`hallyd.coding.Editor._FunctionHandle` attribute), 13
`_abc_impl` (`hallyd.coding.Editor._Handle` attribute), 13
`_abc_impl` (`hallyd.fs._Archive` attribute), 31
`_abc_impl` (`hallyd.fs._TarArchive` attribute), 32
`_abc_impl` (`hallyd.fs._ZipArchive` attribute), 32

`_abc_impl` (*hallyd.fs_monitor.FilesystemMonitor* attribute), 35
`_abc_impl` (*hallyd.fs_monitor.Watcher* attribute), 35
`_abc_impl` (*hallyd.fs_monitor._Watcher* attribute), 37
`_abc_impl` (*hallyd.fs_monitor._Watcher.PollingWatchThread* attribute), 36
`_abc_impl` (*hallyd.fs_monitor._Watcher.WatchThread* attribute), 36
`_abc_impl` (*hallyd.fs_monitor._Watcher.WatchdogWatchThread* attribute), 37
`_abc_impl` (*hallyd.io.Lock* attribute), 38
`_abc_impl` (*hallyd.io._Lock* attribute), 38
`_abc_impl` (*hallyd.ipc.Enabable* attribute), 39
`_abc_impl` (*hallyd.ipc._ThreadedServer* attribute), 41
`_abc_impl` (*hallyd.services.CalendarTask* attribute), 49
`_abc_impl` (*hallyd.services.IntervalTask* attribute), 51
`_abc_impl` (*hallyd.services.NextBootTask* attribute), 52
`_abc_impl` (*hallyd.services.Runnable* attribute), 52
`_abc_impl` (*hallyd.services.Service* attribute), 53
`_abc_impl` (*hallyd.services._CalendarTaskBackend* attribute), 55
`_abc_impl` (*hallyd.services._FunctionRunnable* attribute), 56
`_abc_impl` (*hallyd.services._IntervalTaskBackend* attribute), 56
`_abc_impl` (*hallyd.services._NextBootTaskBackend* attribute), 56
`_abc_impl` (*hallyd.services._ServiceBackend* attribute), 57
`_abc_impl` (*hallyd.services._SystemdBackend* attribute), 59
`_abc_impl` (*hallyd.services._SystemdBackend._Unit* attribute), 58
`_abc_impl` (*hallyd.services._Task* attribute), 60
`_abc_impl` (*hallyd.terminal.AnsiEscapedTextFormatter* attribute), 66
`_abc_impl` (*hallyd.terminal.PlainTextFormatter* attribute), 67
`_abc_impl` (*hallyd.terminal.TextFormatter* attribute), 69
`_abc_impl` (*hallyd.terminal._BareAnsiTerminalColorSpace* attribute), 70
`_abc_impl` (*hallyd.terminal._TerminalColorSpace* attribute), 70
`_abc_impl` (*hallyd.terminal._XTerm256TerminalColorSpace* attribute), 71
`_abc_impl` (*hallyd.typing.SupportsQualifiedName* attribute), 72
`_add_cleanup_task()` (in module *hallyd.cleanup*), 11
`_after_create()` (*hallyd.services.NextBootTaskSetup* method), 52
`_after_create()` (*hallyd.services.ServiceSetup* method), 54
`_after_create()` (*hallyd.services._TaskSetup* method), 61
`_archive()` (in module *hallyd.fs*), 32
`_as_oneshot` (*hallyd.services.ServiceSetup* attribute), 54
`_ast` (*hallyd.coding.Editor* property), 14
`_ast` (*hallyd.coding.Editor._ClassHandle* property), 12
`_ast` (*hallyd.coding.Editor._FunctionHandle* property), 13
`_ast` (*hallyd.coding.Editor._Handle* property), 13
`_before_create()` (*hallyd.services._TaskSetup* method), 61
`_by_qualname()` (in module *hallyd.bundle*), 9
`_calendars` (*hallyd.services.CalendarTaskSetup* attribute), 50
`_changed()` (*hallyd.fs_monitor.FilesystemMonitor* method), 35
`_check_if_changed()` (*hallyd.fs_monitor._Watcher* method), 37
`_create_setup_context()` (in module *hallyd.services*), 63
`_current_cleanup_scope()` (in module *hallyd.cleanup*), 11
`_dependencies` (*hallyd.services._TaskSetup._WithDependencies* attribute), 61
`_description` (*hallyd.services._TaskSetup* attribute), 61
`_deserialize_object_json_loads_object_hook()` (in module *hallyd.bundle*), 9
`_detach_loop_device()` (in module *hallyd.io*), 38
`_discard_output` (*hallyd.services._TaskSetup* attribute), 61
`_disks_sort_key()` (in module *hallyd.disk*), 20
`_do_cleanup()` (in module *hallyd.cleanup*), 11
`_editor` (*hallyd.coding.Editor._Handle* property), 13
`_enabled` (*hallyd.services.ServiceSetup* attribute), 54
`_enum()` (in module *hallyd.bundle*), 9
`_file_name_to_module_name()` (in module *hallyd.py_import*), 48
`_filter_unneeded_dict_entries()` (in module *hallyd.bundle*), 9
`_find_disk_for_setup()` (in module *hallyd.disk*), 20
`_fix_body()` (*hallyd.coding.Editor* method), 14
`_force_changed()` (*hallyd.fs_monitor._Watcher* method), 37
`_function_name_from_body()` (*hallyd.coding.Editor* static method), 14
`_group` (*hallyd.services._TaskSetup* attribute), 61
`_guarded_cleanup()` (in module *hallyd.cleanup*), 11
`_has_path()` (*hallyd.disk.Disk* method), 15
`_have_same_fs_root()` (in module *hallyd.cleanup*), 11
`_interactive` (*hallyd.services.NextBootTaskSetup* attribute), 52
`_intervals` (*hallyd.services.IntervalTaskSetup* attribute), 51
`_is_protocol` (*hallyd.typing.SupportsQualifiedName* attribute), 72

`_is_runtime_protocol` (hallyd.typing.SupportsQualifiedName attribute), 72
`_lsblk()` (in module hallyd.disk), 20
`_name` (hallyd.services._TaskSetup attribute), 61
`_new_bytes()` (in module hallyd.bundle), 10
`_new_datetime_datetime()` (in module hallyd.bundle), 10
`_new_datetime_timedelta()` (in module hallyd.bundle), 10
`_new_functools_partial()` (in module hallyd.bundle), 10
`_new_set()` (in module hallyd.bundle), 10
`_options` (hallyd.services.ServiceSetup attribute), 54
`_post_stop` (hallyd.services.ServiceSetup attribute), 54
`_request()` (hallyd.ipc._Client method), 40
`_request()` (hallyd.ipc._LocalClient method), 40
`_request()` (hallyd.ipc._NetworkClient method), 40
`_restart_delay` (hallyd.services.ServiceSetup attribute), 54
`_runnable` (hallyd.services._TaskSetup attribute), 62
`_serialize_object_json_dumps_default()` (in module hallyd.bundle), 10
`_setup()` (hallyd.fs_monitor._Watcher.WatchThread method), 36
`_setup()` (hallyd.fs_monitor._Watcher.WatchdogWatchThread method), 37
`_short_name()` (hallyd.services._SystemdBackend static method), 60
`_socket_path()` (in module hallyd.ipc), 42
`_start_instantly` (hallyd.services.ServiceSetup attribute), 54
`_start_instantly` (hallyd.services._TimedTaskSetup attribute), 63
`_startup_context` (hallyd.services.ServiceSetup attribute), 54
`_teardown()` (hallyd.fs_monitor._Watcher.WatchThread method), 36
`_teardown()` (hallyd.fs_monitor._Watcher.WatchdogWatchThread method), 37
`_try_process_request()` (hallyd.ipc._ThreadedServer method), 41
`_user` (hallyd.services._TaskSetup attribute), 62
`_wait_event()` (hallyd.fs_monitor._Watcher.PollingWatchThread method), 36
`_wait_event()` (hallyd.fs_monitor._Watcher.WatchThread method), 36
`_watcher` (hallyd.fs_monitor._Watcher.WatchThread property), 36
`_working_dir` (hallyd.services._TaskSetup attribute), 62

A
`AccessDeniedError`, 47
`acquire()` (hallyd.io._Lock method), 38

`acquire()` (hallyd.io.Lock method), 38
`add_answer()` (hallyd.ipc_hub._HubIpObject method), 44
`add_class()` (hallyd.coding.Editor method), 14
`add_cleanup_task()` (in module hallyd.cleanup), 12
`add_decoration()` (hallyd.coding.Editor._WithDecorationSupportHandleMixin method), 13
`add_dependency()` (hallyd.services._TaskSetup._WithDependencies method), 61
`add_function()` (hallyd.coding.Editor method), 14
`add_import()` (hallyd.coding.Editor method), 14
`add_method()` (hallyd.coding.Editor._ClassHandle method), 12
`add_request()` (hallyd.ipc_hub._HubIpObject method), 44
`add_value()` (hallyd.services._SystemdBackend._UnitSpec._MultiDict method), 59
`afterwards` (hallyd.services._TaskSetup._WithDependencies.Dependency attribute), 61
`all_loop_devices()` (in module hallyd.io), 38
`AllAbstractMethodsProvidedByTrick` (class in hallyd.lang), 45
`AnsiEscapedTextFormatter` (class in hallyd.terminal), 66
`answer()` (hallyd.ipc._ThreadedServer._Request method), 41
`answer_request()` (hallyd.ipc_hub.HubWorker method), 43
`append_data()` (hallyd.fs.Path method), 26
`apply_substitutions()` (hallyd.fs.Path method), 26
`as_bare_ansi_terminal_code()` (hallyd.terminal.Color method), 66
`as_html()` (hallyd.terminal.Color method), 66
`as_one_shot()` (hallyd.services.ServiceSetup method), 54
`as_rgb256()` (hallyd.terminal.Color method), 67
`as_rgb_normed()` (hallyd.terminal.Color method), 67
`as_xml` (hallyd.terminal.Text property), 68
`as_xterm256_terminal_code()` (hallyd.terminal.Color method), 67

B
`back_file` (hallyd.io._LoopDevice property), 38
`BadConnectionError`, 39
`by_bare_ansi_terminal_code()` (hallyd.terminal.Color method), 67
`by_gpt_uuid()` (hallyd.disk._PartitionType static method), 19
`by_html()` (hallyd.terminal.Color method), 67
`by_partuuid()` (hallyd.disk.Partition static method), 17
`by_plain_text()` (hallyd.terminal.Text static method), 68

- by_rgb256() (*hallyd.terminal.Color* method), 67
 by_rgb_normed() (*hallyd.terminal.Color* method), 67
 by_uuid() (*hallyd.disk.Partition* static method), 17
 by_xml() (*hallyd.terminal.Text* static method), 68
 by_xterm256_terminal_code() (*hallyd.terminal.Color* method), 67
 byte_size_to_human_readable() (in module *hallyd.fs*), 32
- ## C
- calendar_task() (*hallyd.services._CalendarTaskBackend* method), 55
 calendar_task() (*hallyd.services._SystemdBackend* method), 60
 calendar_task() (in module *hallyd.services*), 63
 CalendarTask (class in *hallyd.services*), 49
 CalendarTaskSetup (class in *hallyd.services*), 49
 CalendarTaskSetup._Calendar (class in *hallyd.services*), 49
 CalendarTaskSetup._DailyCalendar (class in *hallyd.services*), 49
 CalendarTaskSetup._MonthlyCalendar (class in *hallyd.services*), 49
 CalendarTaskSetup._WeeklyCalendar (class in *hallyd.services*), 50
 CalendarTaskSetup._YearlyCalendar (class in *hallyd.services*), 50
 call_now_with_retry() (in module *hallyd.lang*), 46
 change_access() (*hallyd.fs.Path* method), 27
 check_call_with_stdin_string() (in module *hallyd.subprocess*), 65
 check_file_supported_by_begin() (*hallyd.fs._Archive* class method), 31
 check_file_supported_by_begin() (*hallyd.fs._TarArchive* class method), 32
 check_file_supported_by_begin() (*hallyd.fs._ZipArchive* class method), 32
 check_if_changed() (*hallyd.fs_monitor.FilesystemMonitor* method), 35
 check_if_changed() (*hallyd.fs_monitor.FilesystemMonitor._Thread* method), 34
 check_output_with_stdin_string() (in module *hallyd.subprocess*), 65
 class_by_name() (*hallyd.coding.Editor* method), 14
 cleanup_after_exit() (in module *hallyd.cleanup*), 12
 cleanup_task_by_id() (in module *hallyd.cleanup*), 12
 clear_values() (*hallyd.services._SystemdBackend._UnitSpec._MultiDict* method), 59
 client() (in module *hallyd.ipc*), 42
 code (*hallyd.coding.Editor* property), 14
 Color (class in *hallyd.terminal*), 66
 color_to_index() (*hallyd.terminal._BareAnsiTerminalColorSpace* method), 70
 color_to_index() (*hallyd.terminal._TerminalColorSpace* method), 70
 color_to_index() (*hallyd.terminal._XTerm256TerminalColorSpace* method), 71
 combine_disks_to_setups() (in module *hallyd.disk*), 20
 connect_diskimage() (in module *hallyd.io*), 38
 connect_diskimage_buffered() (in module *hallyd.io*), 39
 Connection (class in *hallyd.net*), 47
 Connection.ExecutionResult (class in *hallyd.net*), 47
 CONTINUE_REMOVING_BEHIND_BOUNDARY (*hallyd.fs.OnRemovePassingFileSystemBoundary* attribute), 22
 control_for_bg_color() (*hallyd.terminal.AnsiEscapedTextFormatter* method), 66
 control_for_bg_color() (*hallyd.terminal.PlainTextFormatter* method), 67
 control_for_bg_color() (*hallyd.terminal.TextFormatter* method), 69
 control_for_fg_color() (*hallyd.terminal.AnsiEscapedTextFormatter* method), 66
 control_for_fg_color() (*hallyd.terminal.PlainTextFormatter* method), 67
 control_for_fg_color() (*hallyd.terminal.TextFormatter* method), 69
 control_for_reset() (*hallyd.terminal.AnsiEscapedTextFormatter* method), 66
 control_for_reset() (*hallyd.terminal.PlainTextFormatter* method), 68
 control_for_reset() (*hallyd.terminal.TextFormatter* method), 69
 copy_to() (*hallyd.fs.Path* method), 27
 CouldNotConnectError, 47
 Counter (class in *hallyd.lang*), 45
 create() (*hallyd.disk.RaidSetup* method), 19
 create_calendar_task() (*hallyd.services._CalendarTaskBackend* method), 55
 create_calendar_task() (*hallyd.services._SystemdBackend* method),

60
 create_calendar_task() (in module hallyd.services), 63
 create_diskimage() (in module hallyd.io), 39
 create_interval_task() (hallyd.services._IntervalTaskBackend method), 56
 create_interval_task() (hallyd.services._SystemdBackend method), 60
 create_interval_task() (in module hallyd.services), 63
 create_next_boot_task() (hallyd.services._NextBootTaskBackend method), 56
 create_next_boot_task() (hallyd.services._SystemdBackend method), 60
 create_next_boot_task() (in module hallyd.services), 63
 create_service() (hallyd.services._ServiceBackend method), 57
 create_service() (hallyd.services._SystemdBackend method), 60
 create_service() (in module hallyd.services), 64

D

day (hallyd.services.CalendarTaskSetup._MonthlyCalendar attribute), 50
 day (hallyd.services.CalendarTaskSetup._YearlyCalendar attribute), 50
 day_of_week (hallyd.services.CalendarTaskSetup._WeeklyCalendar attribute), 50
 decorations (hallyd.coding.Editor._WithDecorationSupportHandleMixin property), 13
 description() (hallyd.services._TaskSetup method), 62
 detach() (hallyd.io._LoopDevice method), 38
 dev_path (hallyd.io._LoopDevice property), 38
 dict_from_object() (in module hallyd.bundle), 10
 disable() (hallyd.ipc._ThreadedServer method), 42
 disable() (hallyd.ipc.Enabable method), 40
 disable() (hallyd.services._SystemdBackend._Unit method), 58
 disable() (hallyd.services._Task method), 60
 discard_output() (hallyd.services._TaskSetup method), 62
 Disk (class in hallyd.disk), 14
 disk (hallyd.disk.DiskIntent property), 15
 disk (hallyd.disk.DiskPartition property), 16
 disk_size (hallyd.disk.PartitionSizingEvent property), 18
 disk_space() (in module hallyd.fs), 32
 disk_usage() (in module hallyd.fs), 33
 DiskIntent (class in hallyd.disk), 15
 DiskPartition (class in hallyd.disk), 16
 DiskSetup (class in hallyd.disk), 16
 do_not_restart() (hallyd.services.ServiceSetup method), 54
 do_not_start_instantly() (hallyd.services.ServiceSetup method), 54
 dump() (in module hallyd.bundle), 10
 dumps() (in module hallyd.bundle), 10

E

Editor (class in hallyd.coding), 12
 Editor._ClassHandle (class in hallyd.coding), 12
 Editor._FunctionHandle (class in hallyd.coding), 13
 Editor._Handle (class in hallyd.coding), 13
 Editor._RemovableHandleMixin (class in hallyd.coding), 13
 Editor._WithCodePositionsHandleMixin (class in hallyd.coding), 13
 Editor._WithDecorationSupportHandleMixin (class in hallyd.coding), 13
 effective_partition_setup_order() (in module hallyd.disk), 20
 EFI (hallyd.disk.PartitionTypes attribute), 18
 EfiPartitionSetup() (in module hallyd.disk), 16
 Enabable (class in hallyd.ipc), 39
 enable() (hallyd.ipc._ThreadedServer method), 42
 enable() (hallyd.ipc.Enabable method), 40
 enable() (hallyd.services._SystemdBackend._Unit method), 58
 enable() (hallyd.services._Task method), 60
 end_of_line_position (hallyd.coding.Editor._WithCodePositionsHandleMixin property), 13
 ERROR (hallyd.fs.OnRemovePassingFileSystemBoundary attribute), 22
 error_output (hallyd.net.Connection.ExecutionResult property), 47
 escape() (hallyd.terminal.AnsiEscapedTextFormatter method), 66
 escape() (hallyd.terminal.PlainTextFormatter method), 68
 escape() (hallyd.terminal.TextFormatter method), 69
 exec() (hallyd.net.Connection method), 47
 exec() (hallyd.net.SshConnection method), 48
 execute_in_parallel() (in module hallyd.lang), 46
 expand_archive() (in module hallyd.fs), 33
 expand_archive_to() (hallyd.fs.Path method), 28
 EXT4 (hallyd.disk.PartitionTypes attribute), 18
 extract_all() (hallyd.fs._Archive method), 31
 extract_all() (hallyd.fs._TarArchive method), 32
 extract_all() (hallyd.fs._ZipArchive method), 32

F

FAIL_INSTANTLY (*hallyd.fs.OnRemoveError* attribute),
 22
 FilesystemMonitor (*class in hallyd.fs_monitor*), 34
 FilesystemMonitor._Thread (*class in hal-*
lyd.fs_monitor), 34
 find_disks_for_setups() (*in module hallyd.disk*), 20
 find_partition_for_setup() (*in module hal-*
lyd.disk), 20
 force_changed() (*hal-*
lyd.fs_monitor.FilesystemMonitor method),
 35
 force_changed() (*hal-*
lyd.fs_monitor.FilesystemMonitor._Thread
method), 34
 format() (*hallyd.terminal.Text* method), 68
 free (*hallyd.fs._DiskSpaceInfo* attribute), 32
 fstab_line() (*hallyd.disk.Mountpoint* method), 16
 fstab_type_name (*hallyd.disk._PartitionType* prop-
 erty), 19
 fstype (*hallyd.disk.DiskPartition* property), 16
 full_name (*hallyd.services._SystemdBackend._Unit*
 property), 58

G

get_answer() (*hallyd.ipc_hub.Hub* method), 42
 gpt_uuid (*hallyd.disk._PartitionType* property), 20

H

hallyd
 module, 9
 hallyd._aux
 module, 9
 hallyd._aux.services_helper__call_action
 module, 9
 hallyd._aux.subprocess_helper__call_function
 module, 9
 hallyd.bundle
 module, 9
 hallyd.cleanup
 module, 11
 hallyd.coding
 module, 12
 hallyd.disk
 module, 14
 hallyd.fs
 module, 22
 hallyd.fs_monitor
 module, 34
 hallyd.io
 module, 38
 hallyd.ipc
 module, 39

hallyd.ipc_hub
 module, 42
 hallyd.lang
 module, 45
 hallyd.net
 module, 47
 hallyd.py_import
 module, 48
 hallyd.services
 module, 49
 hallyd.subprocess
 module, 65
 hallyd.terminal
 module, 66
 hallyd.text
 module, 71
 hallyd.typing
 module, 72
 has_key() (*hallyd.terminal.Style* method), 68
 home_dir() (*hallyd.fs.Path* class method), 28
 host_disks() (*in module hallyd.disk*), 21
 host_partition_for_fs_path() (*in module hal-*
lyd.disk), 21
 host_raid_partitions() (*in module hallyd.disk*), 21
 Hub (*class in hallyd.ipc_hub*), 42
 HubWorker (*class in hallyd.ipc_hub*), 43

I

Id (*hallyd.cleanup._CleanupTask* attribute), 11
 id (*hallyd.ipc_hub._HubRequest* property), 45
 import_module() (*in module hallyd.py_import*), 48
 import_types_from_module_dir() (*in module hal-*
lyd.py_import), 48
 indent() (*hallyd.terminal.Text* method), 69
 indentation (*hallyd.coding.Editor* property), 14
 index_to_color() (*hal-*
lyd.terminal._BareAnsiTerminalColorSpace
method), 70
 index_to_color() (*hal-*
lyd.terminal._TerminalColorSpace method),
 70
 index_to_color() (*hal-*
lyd.terminal._XTerm256TerminalColorSpace
method), 71
 install (*hallyd.services._SystemdBackend._UnitSpec*
 property), 59
 interval_task() (*hal-*
lyd.services._IntervalTaskBackend method),
 56
 interval_task() (*hallyd.services._SystemdBackend*
 method), 60
 interval_task() (*in module hallyd.services*), 64
 IntervalTask (*class in hallyd.services*), 51
 IntervalTaskSetup (*class in hallyd.services*), 51

[ipc_path \(hallyd.ipc.IPCServerPathAlreadyExistsError property\), 40](#)
[ipc_path \(hallyd.ipc.IPCServerUnavailableError property\), 40](#)
[IPCServerPathAlreadyExistsError, 40](#)
[IPCServerUnavailableError, 40](#)
[is_active\(\) \(hallyd.services._SystemdBackend._Unit method\), 58](#)
[is_active\(\) \(hallyd.services.Service method\), 53](#)
[is_alive\(\) \(hallyd.net.Connection method\), 47](#)
[is_disk \(hallyd.disk.Disk property\), 15](#)
[is_enabled \(hallyd.ipc._ThreadedServer property\), 42](#)
[is_enabled \(hallyd.ipc.Enabable property\), 40](#)
[is_enabled \(hallyd.services._SystemdBackend._Unit property\), 58](#)
[is_enabled \(hallyd.services._Task property\), 60](#)
[is_interactive_shell\(\) \(in module hallyd.terminal\), 71](#)
[is_process_running\(\) \(in module hallyd.subprocess\), 65](#)
[is_removable \(hallyd.disk.Disk property\), 15](#)
[is_superuser\(\) \(in module hallyd.terminal\), 71](#)
[iterdir\(\) \(hallyd.fs.Path method\), 28](#)

L

[line_wrap_at_maximum_width\(\) \(hallyd.terminal.Text method\), 69](#)
[load\(\) \(in module hallyd.bundle\), 10](#)
[loads\(\) \(in module hallyd.bundle\), 10](#)
[Lock \(class in hallyd.io\), 38](#)
[lock\(\) \(in module hallyd.io\), 39](#)
[locked\(\) \(hallyd.io._Lock method\), 38](#)
[locked\(\) \(hallyd.io.Lock method\), 38](#)
[loop_device_by_dev_path\(\) \(in module hallyd.io\), 39](#)

M

[main\(\) \(in module hallyd._aux.services_helper__call_action\), 9](#)
[main\(\) \(in module hallyd._aux.subprocess_helper__call_function\), 9](#)
[make_dir\(\) \(hallyd.fs.Path method\), 28](#)
[make_file\(\) \(hallyd.fs.Path method\), 29](#)
[make_filesystem\(\) \(hallyd.disk._PartitionSetup method\), 19](#)
[make_filesystem\(\) \(hallyd.disk._PartitionType method\), 20](#)
[mark_current_process_as_cleanup_scope\(\) \(in module hallyd.cleanup\), 12](#)
[match_format_string\(\) \(in module hallyd.text\), 71](#)
[mbr_id \(hallyd.disk._PartitionType property\), 20](#)

[method_by_name\(\) \(hallyd.coding.Editor._ClassHandle method\), 12](#)
[MethodCallErroneousError, 40](#)
[methods \(hallyd.coding.Editor._ClassHandle property\), 13](#)
[module](#)

- [hallyd, 9](#)
- [hallyd._aux, 9](#)
- [hallyd._aux.services_helper__call_action, 9](#)
- [hallyd._aux.subprocess_helper__call_function, 9](#)
- [hallyd.bundle, 9](#)
- [hallyd.cleanup, 11](#)
- [hallyd.coding, 12](#)
- [hallyd.disk, 14](#)
- [hallyd.fs, 22](#)
- [hallyd.fs_monitor, 34](#)
- [hallyd.io, 38](#)
- [hallyd.ipc, 39](#)
- [hallyd.ipc_hub, 42](#)
- [hallyd.lang, 45](#)
- [hallyd.net, 47](#)
- [hallyd.py_import, 48](#)
- [hallyd.services, 49](#)
- [hallyd.subprocess, 65](#)
- [hallyd.terminal, 66](#)
- [hallyd.text, 71](#)
- [hallyd.typing, 72](#)

[month \(hallyd.services.CalendarTaskSetup._YearlyCalendar attribute\), 50](#)
[mount\(\) \(hallyd.disk.Mountpoint method\), 16](#)
[mount\(\) \(hallyd.net.Connection method\), 47](#)
[mount\(\) \(hallyd.net.SshConnection method\), 48](#)
[mount\(\) \(in module hallyd.disk\), 21](#)
[Mountpoint \(class in hallyd.disk\), 16](#)
[mountpoint_spec\(\) \(hallyd.disk._PartitionSetup method\), 19](#)
[move_to\(\) \(hallyd.fs.Path method\), 29](#)

N

[name \(hallyd.coding.Editor._ClassHandle property\), 13](#)
[name \(hallyd.coding.Editor._FunctionHandle property\), 13](#)
[name \(hallyd.services._SystemdBackend._Unit property\), 58](#)
[name \(hallyd.services._Task property\), 60](#)
[name \(hallyd.services._TaskSetup._WithDependencies.Dependency attribute\), 61](#)
[next\(\) \(hallyd.lang.Counter method\), 45](#)
[next_boot_task\(\) \(hallyd.services._NextBootTaskBackend method\), 56](#)

`next_boot_task()` (*hallyd.services._SystemdBackend* method), 60
`next_boot_task()` (in module *hallyd.services*), 64
`NextBootTask` (class in *hallyd.services*), 52
`NextBootTaskSetup` (class in *hallyd.services*), 52
`non_existent()` (*hallyd.fs.Path* method), 30
`NotEfiPartitionSetup()` (in module *hallyd.disk*), 17

O

`object` (*hallyd.ipc._Client* property), 40
`OnRemoveError` (class in *hallyd.fs*), 22
`OnRemovePassingFileSystemBoundary` (class in *hallyd.fs*), 22
`optional` (*hallyd.services._TaskSetup._WithDependencies* attribute), 61
`OrderedPartitionSetupsEntry` (class in *hallyd.disk*), 17
`output` (*hallyd.net.Connection.ExecutionResult* property), 47
`override()` (*hallyd.services._SystemdBackend._Unit* method), 58

P

`part_no` (*hallyd.disk.DiskPartition* property), 16
`part_no` (*hallyd.disk.OrderedPartitionSetupsEntry* property), 17
`Partition` (class in *hallyd.disk*), 17
`partition()` (*hallyd.disk.Disk* method), 15
`partition_path()` (*hallyd.disk.Disk* method), 15
`partition_path()` (in module *hallyd.disk*), 21
`partition_setup` (*hallyd.disk.OrderedPartitionSetupsEntry* property), 17
`partition_tuple()` (in module *hallyd.disk*), 21
`partitions` (*hallyd.disk.Disk* property), 15
`PartitionSetup` (class in *hallyd.disk*), 17
`PartitionSizingEvent` (class in *hallyd.disk*), 18
`PartitionTypes` (class in *hallyd.disk*), 18
`partuuid` (*hallyd.disk.Partition* property), 17
`Path` (class in *hallyd.fs*), 22
`path` (*hallyd.coding.Editor* property), 14
`path` (*hallyd.disk.Disk* property), 15
`path` (*hallyd.disk.Partition* property), 17
`path` (*hallyd.ipc_hub.Hub* property), 43
`paths` (*hallyd.fs_monitor._Watcher* property), 37
`paths` (*hallyd.fs_monitor.FilesystemMonitor* property), 35
`payload` (*hallyd.ipc_hub._HubRequest* property), 45
`pending_request_ids()` (*hallyd.ipc_hub._HubIpcObject* method), 44
`pid_for_process_permanent_id()` (in module *hallyd.subprocess*), 65
`PlainTextFormatter` (class in *hallyd.terminal*), 67

`plug_into_hub()` (*hallyd.ipc_hub.HubWorker* class method), 43
`pop_answer()` (*hallyd.ipc_hub._HubIpcObject* method), 44
`pretty_print_xml()` (in module *hallyd.text*), 71
`process_permanent_id_for_pid()` (in module *hallyd.subprocess*), 65
`put_request()` (*hallyd.ipc_hub.Hub* method), 43

R

`RAID` (*hallyd.disk.PartitionTypes* attribute), 18
`raid_setups_from_disk_intents()` (in module *hallyd.disk*), 22
`RaidPartition` (class in *hallyd.disk*), 18
`RaidPartitionSetup` (class in *hallyd.disk*), 18
`RaidSetup` (class in *hallyd.disk*), 18
`reboot()` (*hallyd.services.Runnable* method), 52
`relative_to()` (*hallyd.fs.Path* method), 30
`release()` (*hallyd.io._Lock* method), 38
`release()` (*hallyd.io.Lock* method), 38
`reload()` (*hallyd.services._SystemdBackend._Unit* method), 59
`reload()` (*hallyd.services.Service* method), 53
`reload_devices()` (in module *hallyd.disk*), 22
`remove()` (*hallyd.cleanup._CleanupTask* method), 11
`remove()` (*hallyd.coding.Editor._RemovableHandleMixin* method), 13
`remove()` (*hallyd.fs.Path* method), 30
`remove_calendar_task()` (*hallyd.services._CalendarTaskBackend* method), 55
`remove_calendar_task()` (*hallyd.services._SystemdBackend* method), 60
`remove_calendar_task()` (in module *hallyd.services*), 64
`remove_decoration()` (*hallyd.coding.Editor._WithDecorationSupportHandleMixin* method), 13
`remove_interval_task()` (*hallyd.services._IntervalTaskBackend* method), 56
`remove_interval_task()` (*hallyd.services._SystemdBackend* method), 60
`remove_interval_task()` (in module *hallyd.services*), 64
`remove_next_boot_action()` (*hallyd.services._NextBootTaskBackend* method), 56
`remove_next_boot_action()` (*hallyd.services._SystemdBackend* method), 60

`remove_next_boot_action()` (in module `hallyd.services`), 64
`remove_service()` (`hallyd.services._ServiceBackend` method), 57
`remove_service()` (`hallyd.services._SystemdBackend` method), 60
`remove_service()` (in module `hallyd.services`), 64
`repartition()` (`hallyd.disk.DiskIntent` method), 15
`request_arrived()` (`hallyd.ipc_hub.HubWorker` method), 43
`request_by_id()` (`hallyd.ipc_hub._HubIpcObject` method), 44
`request_disappeared()` (`hallyd.ipc_hub.HubWorker` method), 44
`request_id` (`hallyd.ipc_hub._HubRequest` property), 45
`request_payload` (`hallyd.ipc_hub._HubRequest` property), 45
`restart()` (`hallyd.services._SystemdBackend._Unit` method), 59
`restart()` (`hallyd.services.Service` method), 53
`returncode` (`hallyd.net.Connection.ExecutionResult` property), 47
`run()` (`hallyd.fs_monitor._Watcher.WatchThread` method), 36
`run()` (`hallyd.fs_monitor.FilesystemMonitor._Thread` method), 34
`run()` (`hallyd.ipc._ThreadedServer._MainThread` method), 41
`run()` (`hallyd.lang._ExecuteParallelThread` method), 45
`run()` (`hallyd.services._FunctionRunnable` method), 56
`run()` (`hallyd.services.Runnable` method), 53
`run_as_user()` (`hallyd.services._TaskSetup` method), 62
`run_in_working_dir()` (`hallyd.services._TaskSetup` method), 62
`run_interactively()` (`hallyd.services.NextBootTaskSetup` method), 52
`Runnable` (class in `hallyd.services`), 52
`Runnable._FinishAndReboot`, 52

S

`schedule_by_interval()` (`hallyd.services.IntervalTaskSetup` method), 51
`schedule_daily()` (`hallyd.services.CalendarTaskSetup` method), 50
`schedule_monthly()` (`hallyd.services.CalendarTaskSetup` method), 50
`schedule_weekly()` (`hallyd.services.CalendarTaskSetup` method), 50
`schedule_yearly()` (`hallyd.services.CalendarTaskSetup` method), 51
`SerializingError`, 9
`Service` (class in `hallyd.services`), 53
`service` (`hallyd.services._SystemdBackend._UnitSpec` property), 59
`service()` (`hallyd.services._ServiceBackend` method), 57
`service()` (`hallyd.services._SystemdBackend` method), 60
`service()` (in module `hallyd.services`), 65
`ServiceSetup` (class in `hallyd.services`), 53
`set_data()` (`hallyd.fs.Path` method), 30
`set_value()` (`hallyd.services._SystemdBackend._UnitSpec._MultiDict` method), 59
`setup` (`hallyd.disk.DiskIntent` property), 15
`size_bytes` (`hallyd.disk.Disk` property), 15
`SKIP_AND_FAIL_LATER` (`hallyd.fs.OnRemoveError` attribute), 22
`SKIP_AND_IGNORE` (`hallyd.fs.OnRemoveError` attribute), 22
`SshConnection` (class in `hallyd.net`), 47
`stable_path` (`hallyd.disk.Disk` property), 15
`stable_path` (`hallyd.disk.Partition` property), 17
`stable_udev_filter()` (`hallyd.disk.Disk` method), 15
`start()` (`hallyd.services._SystemdBackend._Unit` method), 59
`start()` (`hallyd.services.Service` method), 53
`start_function_in_new_process()` (in module `hallyd.subprocess`), 66
`start_instantly()` (`hallyd.services._TimedTaskSetup` method), 63
`starts_at_position` (`hallyd.coding.Editor._WithCodePositionsHandleMixin` property), 13
`stop()` (`hallyd.disk.RaidPartition` method), 18
`stop()` (`hallyd.fs_monitor._Watcher.WatchThread` method), 36
`stop()` (`hallyd.fs_monitor.FilesystemMonitor._Thread` method), 34
`stop()` (`hallyd.ipc._ThreadedServer._MainThread` method), 41
`stop()` (`hallyd.services._SystemdBackend._Unit` method), 59
`stop()` (`hallyd.services.Service` method), 53
`Style` (class in `hallyd.terminal`), 68
`success_required` (`hallyd.services._TaskSetup._WithDependencies.Dependency` attribute), 61
`SupportsQualifiedName` (class in `hallyd.typing`), 72
`SWAP` (`hallyd.disk.PartitionTypes` attribute), 18

T

take_1st_level_default (hallyd.fs._Archive property), 32

take_1st_level_default (hallyd.fs._TarArchive attribute), 32

take_1st_level_default (hallyd.fs._ZipArchive attribute), 32

task_id (hallyd.cleanup._CleanupTask property), 11

temp_dir() (hallyd.fs.Path class method), 31

temp_dir() (in module hallyd.fs), 33

terminal_width() (in module hallyd.terminal), 71

Text (class in hallyd.terminal), 68

TextFormatter (class in hallyd.terminal), 69

threaded_server() (in module hallyd.ipc), 42

time (hallyd.services.CalendarTaskSetup._DailyCalendar attribute), 49

time (hallyd.services.CalendarTaskSetup._MonthlyCalendar attribute), 50

time (hallyd.services.CalendarTaskSetup._WeeklyCalendar attribute), 50

time (hallyd.services.CalendarTaskSetup._YearlyCalendar attribute), 50

timer (hallyd.services._SystemdBackend._UnitSpec property), 59

total (hallyd.fs._DiskSpaceInfo attribute), 32

try_create() (hallyd.fs_monitor._Watcher.WatchThread class method), 36

try_lock_request() (hallyd.ipc_hub.HubWorker method), 44

TRY_UNMOUNTING (hallyd.fs.OnRemovePassingFileSystemBoundary attribute), 22

TRY_UNMOUNTING_FORCEFULLY (hallyd.fs.OnRemovePassingFileSystemBoundary attribute), 22

types_from_module() (in module hallyd.py_import), 49

U

udev_rule_for_alias() (hallyd.disk.DiskIntent method), 16

umount() (hallyd.disk.Mountpoint method), 17

umount() (hallyd.net.Connection method), 47

umount() (hallyd.net.SshConnection method), 48

umount() (in module hallyd.disk), 22

unique_id() (in module hallyd.lang), 46

unit (hallyd.services._SystemdBackend._UnitSpec property), 59

unit_type (hallyd.services._SystemdBackend._Unit property), 59

UNUSED (hallyd.disk.PartitionTypes attribute), 18

used (hallyd.fs._DiskSpaceInfo attribute), 32

uuid (hallyd.disk.Partition property), 17

V

value() (hallyd.terminal.Style method), 68

verify_tool_available() (in module hallyd.subprocess), 66

W

wait_changed() (hallyd.fs_monitor._Watcher method), 37

wait_changed() (hallyd.fs_monitor.Watcher method), 35

watch() (in module hallyd.fs_monitor), 37

Watcher (class in hallyd.fs_monitor), 35

with_friendly_repr_implementation() (in module hallyd.lang), 46

with_option() (hallyd.services.ServiceSetup method), 55

with_post_stop_command() (hallyd.services.ServiceSetup method), 55

with_restart_delay() (hallyd.services.ServiceSetup method), 55

with_retry() (in module hallyd.lang), 46

with_startup_context() (hallyd.services.ServiceSetup method), 55