Report on File System Control Functions for Oscilloscope

Overview

The provided Python functions facilitate file system management on an oscilloscope (e.g., Tektronix DPO4000 series) through SCPI commands. These methods enable directory creation, file deletion, disk space querying, directory removal, and file renaming directly on the oscilloscope's internal or attached storage. The functions interact with the instrument via a communication interface represented by `self.inst`, which sends SCPI commands and queries.

---

1. `create_directory(self, path: str)`

Purpose: Creates a new directory on the oscilloscope's file system if it does not already exist.

Functionality:

- Extracts the last directory name from the full path using Python's `os.path.basename`.
- Determines the parent directory path using `os.path.dirname`.
- Changes the oscilloscope's current working directory (CWD) to the parent directory.
- Queries the current directory listing (`FILESystem:DIR?`) and parses it to check if the target directory already exists.
- If the directory is absent, sends the `FILESystem:MKDir` SCPI command to create it.
- Prints informative messages about the operation status.

Usage Notes:

- Ensures no redundant directory creation.
- Handles paths with trailing slashes gracefully.
- Assumes the parent directory exists; no explicit error handling for invalid paths.

---

2. `delete_file(self, path: str)`

Purpose: Deletes a specified file from the oscilloscope's file system.

Functionality:

- Sends the SCPI command `FILESystem:DELete` with the file path.
- Prints confirmation upon deletion.

Usage Notes:

- Assumes the file exists; no explicit error checking.
- Useful for cleaning up files generated during measurements or data transfers.

---

3. `disk_freespace(self, disk: str = "C:/") -> int`

Purpose: Retrieves the available free space on a specified disk or partition of the oscilloscope's file system.

Functionality:

- Changes the working directory to the specified disk root.
- Queries free space using `FILESystem:FREESpace?`.
- Returns the free space as a string (ideally should convert to `int` for numerical use).

Usage Notes:

- Default disk is `"C:/"`, but can be customized (e.g., `"D:/"`).
- The returned value represents free space in bytes.
- Current implementation returns a string; converting to integer is recommended for calculations.

---

4. `remove_directory(self, path: str)`

Purpose: Removes (deletes) a directory from the oscilloscope's file system.

Functionality:

- Sends the SCPI command `FILESystem:RMDir` with the directory path.
- Prints confirmation upon deletion.

Usage Notes:

- Assumes the directory is empty or the oscilloscope supports recursive removal.
- No error handling for non-existent or non-empty directories.

---

5. `rename_file(self, old_path: str, new_path: str)`

Purpose: Renames or moves a file within the oscilloscope's file system.

Functionality:

- Sends the SCPI command `FILESystem:RENname` with the old and new file paths.
- Prints confirmation of the rename operation.

Usage Notes:

- Can be used to organize files or change file names after acquisition.
- No explicit error handling for invalid paths or name conflicts.

General Remarks

- **SCPI Command Usage:** All functions rely on standard SCPI commands defined for the oscilloscope's file system interface, ensuring compatibility with supported instruments.
- **Communication Interface:** The `self.inst` object is expected to be a communication interface (e.g., VISA instrument handle) capable of sending commands (`write`) and querying responses (`query`).
- **Error Handling:** The current implementations print status messages but lack robust error handling. Adding try-except blocks and validating responses would improve reliability.

- **Path Handling:** Functions use Python's `os.path` utilities for path parsing, ensuring cross-platform path manipulation before sending SCPI commands with forward slashes as required by the instrument.
- **Extensibility:** These functions can be extended to include:
    - Recursive directory operations,
    - File transfer (upload/download),
    - Detailed error reporting,
    - Support for relative paths and environment variables.

Conclusion

The provided file system control functions offer essential capabilities for managing files and directories on an oscilloscope's internal or external storage via SCPI commands. They enable automation of common tasks such as directory creation, file deletion, disk space monitoring, directory removal, and file renaming, which are critical for efficient data acquisition workflows and instrument file management.

For production use, enhancing error handling and input validation is recommended to ensure robustness in diverse operating conditions.