Write a program to design a File System.

```
import os
import time
from collections import defaultdict

class File:
        def __init__(self, name, size, creation_time=None, data=None):
        self.name = name
        self.size = size
        self.creation_time = creation_time or time.time()
        self.data = data

class Directory:
        def __init__(self, name, parent_dir=None):
        self.name = name
        self.parent_dir = parent_dir
        self.files = []
        self.subdirectories = []

class FileSystem:
        def __init__(self, disk_size):
        self.root_directory = Directory("/")
        self.disk_size = disk_size
        self.free_clusters = set(range(1, disk_size + 1))
        self.file_data = defaultdict(list)  # Store file data in clusters

        def create_file(self, path, size):
        directory, file_name = self._get_directory_and_file_name(path)
        if directory is None:
        print("Directory not found.")
        return
        if size > len(self.free_clusters):
        print("Insufficient disk space.")
        return
        new_file = File(file_name, size)
        new_file.data = [self.free_clusters.pop() for _ in range(size)]
        self.file_data[new_file.name] = new_file.data
        directory.files.append(new_file)
        print(f"File {file_name} created successfully.")

        def delete_file(self, path):
        directory, file_name = self._get_directory_and_file_name(path)
        if directory is None or file_name not in directory.files:
        print("File not found.")
        return
        file_data = self.file_data.pop(file_name)
        self.free_clusters.update(file_data)
        directory.files.remove(file_name)
        print(f"File {file_name} deleted successfully.")

        def read_file(self, path):
        directory, file_name = self._get_directory_and_file_name(path)
        if directory is None or file_name not in directory.files:
```

```
        print("File not found.")
        return
        file_data = self.file_data[file_name]
        return "".join(chr(byte) for byte in file_data)  # Assuming ASCII or similar encoding

        def write_file(self, path, data):
        directory, file_name = self._get_directory_and_file_name(path)
        if directory is None:
        print("Directory not found.")
        return
        file_data = self.file_data.get(file_name)
        if file_data is None:
        print("File not found.")
        return
        if len(data) > len(file_data):
        required_clusters = len(data) - len(file_data)
        if required_clusters > len(self.free_clusters):
                print("Insufficient disk space.")
                return
        file_data.extend(self.free_clusters.pop() for _ in range(required_clusters))
        elif len(data) < len(file_data):
        del file_data[len(data):]
        self.free_clusters.update(file_data[len(data):])
        file_data[:len(data)] = [ord(char) for char in data]
        self.file_data[file_name] = file_data
        print(f"File {file_name} written successfully.")

        # ... other methods (create_directory, list_directory, etc.)

# Example usage
file_system = FileSystem(1024)  # Simulate a 1024-byte disk
file_system.create_directory("/documents")
file_system.create_file("/documents/report.txt", 512)
file_system.write_file("/documents/report.txt", "Hello, world!")
content = file_system.read_file("/documents/report.txt")
print(content)
```