# LU11.L01: Alte Logs

## Herausforderung beim Testen

Beim Testen müssten wir mehrere Tage warten, bis das Logfile das gewünschte Alter erreicht hat. Als Alternative können wir:

- Das Systemdatum in die Zukunft stellen
- Das Erstelldatum der Datei verändern.

Um das Erstelldatum einer Datei zu verändern, benötigen Sie Powershell im Admin-Modus. Der Befehl

```
(Get-Item "C:\log\message.log").CreationTime = "2024-01-10 08:00:00"
```

Setzt das Erstelldatum unserer Datei auf den 10. Januar 2024

main.py

```
'''
module to archive and manage log files

For testing purposes, the log file path is set to C:\log\message.log on
Windows and ~/log/message.log on Linux.
You can change the create time of a file using the following command in
PowerShell (admin mode):
(Get-Item 'C:\log\messagelog').CreationTime = 'YYYY-MM-DD HH:MM:SS'

'''
import argparse
import os
import sys
import time
from datetime import datetime, timedelta

def main():
    args = parse_arguments()
    log_file_path = get_log_file_path()
    log_dir = os.path.dirname(log_file_path)

    file_age_days = get_file_age_in_days(log_file_path)
    if file_age_days is None:
        print(f'Log file does not exist: {log_file_path}')
        sys.exit(1)

    if file_age_days >= args.max_age_days:
        backup_file_path = archive_log_file(log_file_path)
        print(f'Archived log file to: {backup_file_path}')
        manage_backups(log_dir, args.max_backups)
```

```
    else:
        print('No action needed; log file is not old enough to
archive.')

def parse_arguments():
    parser = argparse.ArgumentParser(description='Archive and manage
log files.')
    parser.add_argument('max_age_days', type=int, help='Maximum age of
the log file in days before archiving.')
    parser.add_argument('max_backups', type=int, help='Number of backup
files to retain.')
    return parser.parse_args()

def get_log_file_path():
    if os.name == 'nt':  # Windows
        return r'C:\log\message.log'
    else:  # Linux and others
        return os.path.expanduser('~/log/message.log')

def get_file_age_in_days(file_path):
    '''
    Get the age of the file in days.
    '''
    if not os.path.exists(file_path):
        return None
    file_mod_time = os.path.getctime(file_path)
    file_age = datetime.now() - datetime.fromtimestamp(file_mod_time)
    return file_age.days

def archive_log_file(log_file_path):
    '''
    create a backup of the log file and clear the log file
    '''
    today_str = datetime.now().strftime('%Y%m%d')
    backup_file_path = f'{log_file_path.rsplit(".",
1)[0]}{today_str}.log'
    os.rename(log_file_path, backup_file_path)
    open(log_file_path, 'w').close()
    return backup_file_path

def manage_backups(log_dir, max_backups):
    '''
    Manage the number of backup files.
    '''
    backups = [f for f in os.listdir(log_dir) if
f.startswith('message') and f.endswith('.log') and f != 'message.log']
    backups.sort()
    while len(backups) > max_backups:
        oldest_backup = backups.pop(0)
```

```
        os.remove(os.path.join(log_dir, oldest_backup))
        print(f'Removed log file: {oldest_backup}')



if __name__ == '__main__':
    main()
```

---

M122-LU11

Marcel Suter