Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет»

Институт математики и информационных технологий

Рыбин Егор Ильич

Отчет по дисциплине «Верификация программного обеспечения»

Петрозаводск — 2019

Содержание

1 Объект тестирования

1.1 Описание приложения

Приложение для создания бекапов. На основе конфигурационного файла создается локальная копия другого локального или удаленного файла. Доступ к удаленным файлам осуществляется через sftp. Дополнительно возможно gzip сжатие получающихся бекапов.

Программа написана на языке программирования Go. Для создания тестов используется встроенный модуль testing.

1.2 Функции приложения

Функции приложения:

1. Создание бекапа
   (a) Локального файла
   (b) Удаленного файла через sftp
2. Сжатие бекапа

1.3 Архитектура программы

Модули:

1. Основной модуль
   - Чтение конфигурационного файла
   - Подготовка и вызов соответствующих функций
   - Копирование файла
2. File
   Работа с локальными файлами.
   - Открыте
   - Чтение
3. Sftp
   Работа с удаленными файлами через sftp.
   - Открыте
   - Чтение
4. Gzip
   Добавление сжатия на поток записи файла.

Рис. 1: Архитектура программы.

2 Подготовка к тестированию

Для проведения полноценного тестирования требуются заранее подготовленные локальные файлы, удаленные файлы с доступом по sftp и тестовый сервер.

Для создания тестового ssh сервера с файлами, с которых можно будет создавать бекапы, используется docker. За основу взят docker image `hermsi/alpine-sshd` – контейнер основанный на Alpine Linux с установленным sshd демоном и возможностью очень просто указать необходимые для авторизации логин и пароль, что необходимо для упрощения проведения процедуры тестирования.

Для ssh подключения. В качестве пользователя используется аккаунт `root`. В качестве пароля задано `pass1234`. Сервис слушает на порту под номером `1234` на `localhost`

На сервере в домашем каталоге пользователя `root` создан файл:

- `1-ssh.txt` и содержанием '`Hello, sftp!`'

Локально кампьютере, в дирректории из которой происходит процесс тестирования, созданы файлы:

- `test.txt` с содержимым вида '`Blah, blah blah. Blah! Blah blah blah, blah.`'
- `t/test.txt` с содержанием вида '`Test1`' и символическая ссылка `t2` на папку `t`

Так же в тестирование задействованы стандартные для unix-like систем, уже существующие файлы:

- `/dev/zero` специальный устройство генерирующее поток `0`, к которому можно обращатся как к обычному файлу;
- `/etc/shadow` файл к которому у обычного пользователя нет доступа;

Все файлы которые будут произведенны во время тестирования (созданные бекапы файлов) будут находится либо в каталоге из которого проводилось тестирование, если для теста необходимо убедится в правильности содержимымого файла, или во временном каталоге `/tmp/`.

3 Тестирование

3.1 Блочное тестирование

Блочные тесты проверяют работоспособность отдельных функций программы. Их поведение не зависит от результатов работы других тестируемых функций. Для их проверки в качестве аргументов при вызове функции передаются различные значения, после чего возвращаемый результат функции проверяется на наличие ошибок.

Тестируемые функции:

- `OpenSftpFile(user, password, host, file string) (*Sftp, error)` – Открытие удаленного файла через sftp.
  В качестве аргументов принимает имя, пароль которые используются в ssh, ssh адресс, к которому необходимо подключаться, и имя файла.
  - user – строка с именем пользователя;

  - passwod – строка с парелем пользователя;
  - host – строка с хостом для подключения;
  - file – строка с путем до файла на хосте;

  Возвращает структуру с возможностью чтения удалленого фалйа по sftp.

- `OpenLocalFile(src string) (*File, error)` – Открыте локального файла на чтение.
  В качестве аргумента принимает имя файла.

  - src – строка с путем до файла на локальном компьютере;

  Вовзращает структуру с возможностью чтения локального файла.

- `AddGzipIf(src io.WriteCloser, is bool) io.WriteCloser` – Добавление сжатия.
  В качестве аргументов принимает поток записи и булеву переменную.

  - src – поток записи бекап файла;
  - is – булева переменная, обозначающая нужно ли добалять поток сжатия;

  Возвращает модифицированный поток записи, с добавленным сжатием.

- `Copy(src io.Reader, dst io.Writer) (int64, error)` – Копирование.
  В качестве аргументов принимаются поток чтения и записи.

  - src – поток чтения файла с которго делается бекап;
  - dst – поток записи файла в который записывается бекап;

  Копирует данные из структуру с возможностью чтения, переданной в первом аргументе, в структуру с возможностью записи, переданную во втором аргументе.

3.2 Интеграционное тестирование

При интеграционном тестировании проверяется взаимодействие существтующих модулей. Описанные выше, в блочном тестирование, функции, принимают ввод и отдают вывод другим. Таким образом проверяется отсутсвие ошибок во взаимодействие их между друг другом.

Для интерграционного тестирования используется функция:

- `runInstance(data backupInstance) error` – Создание бекапа.
  В качестве аргументов передается структура содержащая необходимую для осуществления бекапа информацию: имя, пароль, хост если это sftp, имя файла, имя сохраненной копии файла.
  Возвращает ошибку, в случае возникновения соответсвующей во время выполнения функции.

Она, на основе входной структуры, вызывает соответсвующие другие функции, позволяя протестировать сценарии задействущие несколько функций:

- OpenLocalFile → Copy
- OpenSftpFile → Copy
- OpenLocalFile → AddGzipIf → Copy

3.3 Аттестационное тестирование

Тестирование работы программы в целом. Произовдится путем запуска тестировщиком скомпилированной программы и созданием необходимых конфигурационых файлов. Таким образом проверяется работоспособность программы в виде приближенном к реальным условиям эксплуатации.

Запуск программы производится из консоли посредством команды:

```
./main conifg.json
```

– где `main` это название скомпилированной программы, а
`config.json` это название конфигурационного файла.

Для общего тестирования программы подготовленны специальные файлы конфигурации, которые позволяют протестировать работы программы в целом, сценарии вида:

- Start → ParseConfig → OpenLocalFile → Copy → End
- Start → ParseConfig → OpenSftpFile → AddGzipIf → Copy → End

3.4 Нагрузочное тестирование

Нагрузочное тестирование проверяет работу приложения в экстримальных условиях: при большом количестве операций копирования, при копировании больших файлов.

3.5 Критерии прохождения тестирования

Тест считается пройденным если полученный и ожидаемый результат совпадают. Тестирование считается пройденным если во время его прохождения не выявлено критических ошибок и процент пройденных тестов не меньше 80%.

3.6 Критерии остановки тестирования

Тестирование должно быть остановленно если количество непройденных тестов больше 30% от общего количества, а так же при обнаружении критических ошибок сильно влияющих на функциональность приложения.

3.7 Критерии возобновления тестирования

Тестирование возобнавляется при исправлении ошибок на предыдущем этапе тестирования.

3.8 Описание тестов

3.8.1 Блочные

№	1
Функция	OpenSftpFile
Тип	Позитивный
Входные данные	user: root password: pass1234 host: localhost:1234 file: 1-ssh.txt
Ожидаемый результат	sftp авторизация успешно пройдена, файл существует на сервере и успешно открыт

Таблица 1: Тестирование функции OpenSftpFile.

№	2
Функция	OpenSftpFile
Тип	Негативный
Входные данные	user: root password: wrong host: localhost:1234 file: 1-ssh.txt
Ожидаемый результат	Ошибка: sftp авторизация не пройдена, файл не открыт

Таблица 2: Тестирование функции OpenSftpFile.

№	3
Функция	OpenSftpFile
Тип	Негативный
Входные данные	user: root password: pass1234 host: localhost:1234 file: wrong.txt
Ожидаемый результат	Ошибка: sftp авторизация успешно пройдена, файл не существует на сервере и не будет открыт

Таблица 3: Тестирование функции OpenSftpFile.

№	4
Функция	OpenLocalFile
Тип	Позитивный
Входные данные	file: test.txt
Ожидаемый результат	локальный файл существует и успешно открыт

Таблица 4: Тестирование функции OpenLocalFile.

№	5
Функция	OpenLocalFile
Тип	Негативный
Входные данные	file: notexist.txt
Ожидаемый результат	Ошибка: локальный файл не существует и не открыт

Таблица 5: Тестирование функции OpenLocalFile.

№	6
Функция	AddGzipIf
Тип	Позитивный
Входные данные	src: *stream is: true
Ожидаемый результат	добавление сжатия на поток

Таблица 6: Тестирование функции AddGzipIf.

№	7
Функция	AddGzipIf
Тип	Позитивный
Входные данные	src: *stream is: false
Ожидаемый результат	не добавление сжатия на поток

Таблица 7: Тестирование функции AddGzipIf.

№	8
Функция	Copy
Тип	Позитивный
Входные данные	src: *ReadStream dst: *WriteStream
Ожидаемый результат	данные скопированны из src в dst

Таблица 8: Тестирование функции Copy.

№	9
Функция	OpenLocalFile
Тип	Позитивный
Входные данные	file: /dev/zero
Ожидаемый результат	Файл не будет открыт на чтение тк является специальным устройством

Таблица 9: Тестирование функции OpenLocalFile.

№	10
Функция	OpenLocalFile
Тип	Позитивный
Входные данные	file: /etc/shadow
Ожидаемый результат	локальный файл существует, но не может быть открыт из за отсутсвующих прав доступа

Таблица 10: Тестирование функции OpenLocalFile.

№	11
Функция	OpenLocalFile
Тип	Позитивный
Входные данные	file: t2/test.txt
Ожидаемый результат	t2 является символической ссылкой на другую дирке-торию где локальный файл существует, файл должен быть успешно открыт

Таблица 11: Тестирование функции OpenLocalFile.

3.8.2 Интеграционные

№	101
Функция	runInstance
Тип	Позитивный

Входные данные	data: {
	type: file
	from: test.txt
	to: test.txt_backup
	gzip: false
	}

Ожидаемый результат	Копирование локального файла test.txt в test.txt_backup

Таблица 12: Бекап локального файла.

№	102
Функция	runInstance
Тип	Позитивный

Входные данные	data: {
	user: root
	pass: pass1234
	host: localhost:1234
	from: 1-ssh.txt
	to: 1-ssh.txt_backup
	gzip: false
	}

Ожидаемый результат	Копирование файла удаленного файла 1-ssh.txt через sftp в 1-ssh.txt_backup

Таблица 13: Бекап удаленного файла.

№	103
Функция	runInstance
Тип	Позитивный

Входные данные	data: {
	type: file
	from: test.txt
	to: test.txt_backup
	gzip: true
	}

Ожидаемый результат	Копирование локального файла test.txt в test.txt_backup с добавлением сжатия

Таблица 14: Бекап со сжатием локального файла.

3.8.3 Аттестационные

№	201
Функционал	Возможность создавать бекапы локальных файлов
Исходные данные	запущена программа из командной строки
Тип	Позитивный
Содержимое конфигурационного файла	{ "backup": [{ "type": "file "from": "test.txt "to": "backup_test.txt" }] }
Запускающая команда	`./main test201.json`
Ожидаемый результат	прочтен файл конфигурация, сделан локального бекап файла с именем `test.txt` в файл `backup_test.txt`

Таблица 15: Запуск программы для бекапа однго локального файла.

№	202
Функционал	Возможность создавать бекапы локальных файлов и сжимать их
Исходные данные	запущена программа из командной строки
Тип	Позитивный
Содержимое конфигурационного файла	{ "backup": [{ "type": "file "from": "test.txt "gzip": true, "to": "backup_test.txt" }] }
Запускающая команда	`./main test202.json`
Ожидаемый результат	прочтен файл конфигурация, сделан локального бекап файла с именем `test.txt` в файл `backup_test.txt`, файла бекапа сжат с использованием gzip.

Таблица 16: Запуск программы для бекапа со сжатием одного локального файла.

№	203
Функционал	Возможность создания бекапов удаленных файлов через sftp
Исходные данные	запущена программа из командной строки
Тип	Позитивный

Содержимое конфигурационного файла	{ "backup": [
	
	{ "type": "file
	"from": "test.txt
	"to": "backup_test.txt"
	},
	{
	"type": "sftp
	"from": "1-ssh.txt
	"to": "1-ssh.txt_backup
	"pass": "pass1234
	"user": "root
	"host": "localhost:1234"
	}
	]
	}

Запускающая команда	`./main test203.json`
Ожидаемый результат	прочтен файл конфигурация, сделан бекап локального файла с именем `test.txt` в файл `backup_test.txt` И так же удаленного файла `1-ssh.txt` по sftp в файл `1-ssh.txt_backup`

Таблица 17: Запуск программы для бекапа нескольких файлов.

№	204
Функционал	Корректное завершение при отсутсвие указанния конфигурационного файла
Исходные данные	запущена программа из командной строки
Тип	Негативный
Содержимое конфигурационного файла	нет
Запускающая команда	`./main`
Ожидаемый результат	Ошибка: файл конфигурации не указан, прекращение работы программы

Таблица 18: Запуск программы без конфигурационного файла.

№	205
Функционал	Корректное завершение при некорректном конфигурационном файле
Исходные данные	запущена программа из командной строки
Тип	Негативный
Содержимое конфигурационного файла	{ "backup": [{ "type": "file "from": "test.txt" }] }
Запускающая команда	`./main test205.json`
Ожидаемый результат	Ошибка: прочтен файл конфигурации, в файле пропущено поле 'to' указывающее куда делать бекап.

Таблица 19: Запуск программы для бекапа со сжатием одного локального файла.

3.8.4 Нагрузочное тестирование

№	301
Входные данные	запущена программа из командной строки, конфигурационный файл составлн таким образом что требует 10 подключений по sftp.
Тип	Позитивный
Запускающая команд	`time ./main test301.json`
Ожидаемый результат	прочтен файл конфигурация, сделан бекап указанных в конфигурации файлов.

Таблица 20: Запуск программы с конфигурацией требующей большого количества подключений.

№	302
Исходные данные	запущена программа из командной строки
Тип	Позитивный
Содержимое конфигурационного файла	{ "backup": [{ "type": "file "from": "10G "to": "/dev/null" }] }
Запускающая команда	`./main test302.json`
Ожидаемый результат	прочтен файл конфигурация, файл размером 10 гигабайт успешно скопирован

Таблица 21: Запуск программы для копирования из специальных файлов.

3.9 Трассируемость требований

Требования к функциональности приложения:

1. Создание бекапа
   (a) Локального файла
   (b) Удаленного файла через sftp
2. Сжатие бекапа

№ теста	1.а	1.б	2
1		+	
2		+	
3		+	
4	+		
5	+		
6			+
7			+
8	+	+	
9	+		
10	+		
11	+		
101	+		
102		+	
103	+		+
201	+		
202	+		+
203		+	
204	+	+	+
205	+	+	+
301		+	
302	+		

Таблица 22: Трассируемость требований.

3.10 Примеры тестов

Тест проверки работоспособности функции открытия удаленного файла:

```
func TestOpenSftp(t *testing.T) {
    _, err := OpenSftpFile("root", "pass1234", "localhost:1234", "1-ssh.txt")
    if err != nil {
        t.Error("Sftp file failed to open")
    }
}
```

Тест запуска необходимых функций на основе структуры – копирование удаленного файла.

```
func TestRunSftp(t *testing.T) {
    b := backupInstance{
        Type: "sftp",
        From: "1-ssh.txt",
        To: "1/10-ssh.txt",
        Pass: "pass1234",
        User: "root",
        Host: "localhost:1234",
    }

    err := runInstance(b)
    if err != nil { t.Error("Can't copy:", err) }
}
```

4 Результаты тестирования

4.1 Отчет о проведение тестирования

№ теста	Дата	Результат
1	18.12.2019	Пройден
2	18.12.2019	Пройден
3	18.12.2019	Пройден
4	18.12.2019	Пройден
5	18.12.2019	Пройден
6	18.12.2019	Пройден
7	18.12.2019	Пройден
8	18.12.2019	Пройден
9	18.12.2019	Пройден
10	18.12.2019	Пройден
11	18.12.2019	Ошибка (Отчет №1)
101	18.12.2019	Пройден
102	18.12.2019	Пройден
103	18.12.2019	Пройден
201	18.12.2019	Пройден
202	18.12.2019	Пройден
203	18.12.2019	Ошибка (Отчет №2)
204	18.12.2019	Пройден
205	18.12.2019	Пройден
301	18.12.2019	Ошибка (Отчет №3)
302	18.12.2019	Пройден

Таблица 23: Результаты проведения тестирования.

4.2 Отчет об ошибках

- Отчет №1:
  Тест: 11

Описание теста: Таблица 11
Описание ошибки: Ошибка при попытке создания бекапа локального файла файла:

```
--- FAIL: TestOpenLocalThrouLink (0.00s)
    test_test.go:49: Local file failed to open: stat t2/test302: no such file
```

Ожидаемый результат: Создание бекапа локального файла указаного через символическую ссылку.
Полученный результат: Ошибка.

- Отчет №2:
  Тест: 203
  Описание теста: Таблица 17
  Описание ошибки: Ошибка при попытке создания бекапа удаленного файла:

```
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
```

Ожидаемый результат: Создание бекапа удаленного файла через sftp.
Полученный результат: Ошибка.

- Отчет №2:
  Тест: 301
  Описание теста: Таблица 20
  Описание ошибки: Ошибка при попытке создания бекапа удаленного файла:

```
Copy sftp from 1-ssh.txt to 1/1-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/2-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/3-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/4-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/5-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/6-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
```

```
Copy sftp from 1-ssh.txt to 1/7-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/8-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/9-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
Copy sftp from 1-ssh.txt to 1/10-ssh.txt
Connecting to root  localhost:1234 1-ssh.txt
Error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [
```

Ожидаемый результат: Создание бекапа 10 (десяти) удаленных файлов через sftp.
Полученный результат: Ошибка.

4.3 Результаты

В ходе тестирования были выявленны 3 ошибки. Одна была выявленна в рамках блочного тестирования, еще одна в рамках аттестационного, и еще одна в рамках нагрузочного. Ошибки описанные в Отчете №2 и Отчете №3 на деле являются результатом одной ошибки. Часть из найденных ошибок существенно влияют на работу системы, но могут быть быстро исправленны без больших затрат.

После исправления найденных ошибок программа должна стать работоспособной.