ASTERCONF
- 2020

Что под капотом у Asterisk: Архитектура, модульность, шина, потоки

Доклад подготовил Штейнлихт Олег

01

Архитектура Asterisk

Сервер
asterisk

Клиент
asterisk -r

взаимодействие по unix-сокету

CLI на базе библиотеки libedit

Зависимости

- libedit - обеспечивает поддержку работы с терминалом (командная строка, история команд)
- libjansson - поддержка формата json
- openssl - поддержка протокола tls

CORE		INTERFACE	BUSINESS
static	loadable		
/main pbx.c channel.c bridge.c bribge_channel. translate.c cdr.c dial.c cli.c frame.c utils.c stasis.c threadpool.c ...	/res res_pjsip.c res_pjsip_* res_dahdi.c res_agi.c res_ari.c res_odbc.c res_snmp.c res_srtp.c res_xmpp.c res_realtime.c res_sendtext.c ... /formats format_pcm.c /codecs codec_alaw.c	/channels chan_sip.c chan_pjsip.c chan_iax2.c	/apps app_dial.c app_queue.c app_confbridge.c app_playback.c app_verbose.c app_echo.c app_cdr.c app_sendtext.c ... /func func_channel.c func_curl.c func_pjsip_endpoint.c

Разработка нового приложения

- Добавить модуль res_modulename.c в папку res (это надо делать, если необходимой функциональности еще нет)
- Добавить модуль app_modulename.c в папку apps (или funcs, если мы разрабатываем функцию)

Сборщик автоматически определит ваше приложение и его можно будет собрать.
Сборка:

```
# make
```

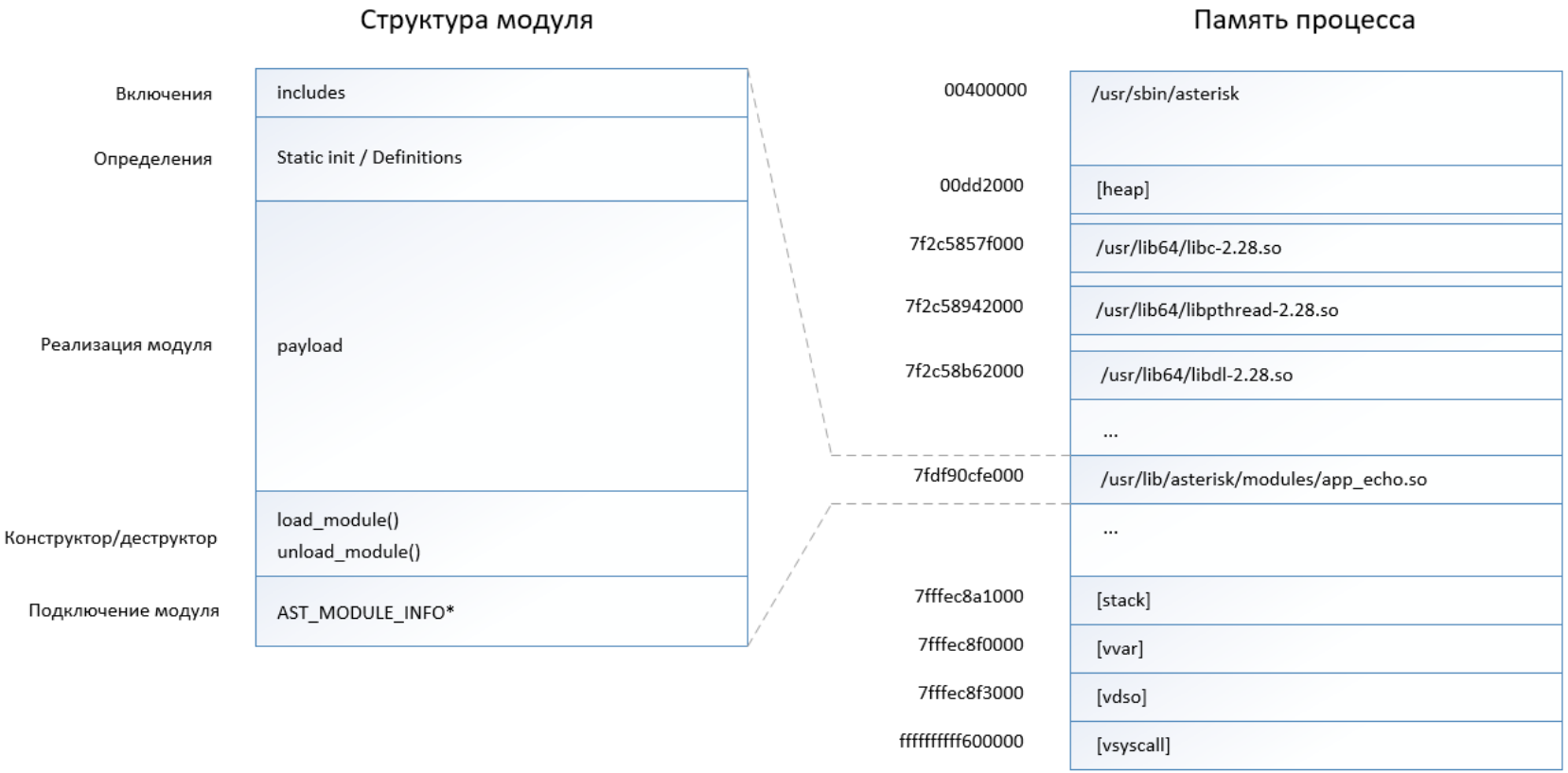
Установка:

```
# make upgrade
```

Модули

Модуль - это необязательная часть программы. Модуль может быть скомпонован с программой статически (во время компиляции) или динамически (во время выполнения).
В линуксе, динамически загружаемые модули реализуются на базе **shared object** (.so) файлов

- dlopen() - загрузка so-файла. Модуль загружается в адресное пространство процесса
- dlclose() - закрытие so-файла. Модуль выгружается из адресного пространства, память освобождается
- dlsym() - поиск символа, в загруженом so-файле (в астериске не используется)



cat /proc/<pid>/maps

Asterisk module.h

```
struct ast_module_info {
	struct ast_module *self;
	enum ast_module_load_result (*load)(void);
	int (*reload)(void);
	int (*unload)(void);
	const char *name;
	const char *description;
	const char *key;
	unsigned int flags;
	const char buildopt_sum[33];
	unsigned char load_pri;
	const char *requires;
	const char *optional_modules;
	const char *enhances;
	void *reserved1;
	void *reserved2;
	void *reserved3;
	void *reserved4;
	enum ast_module_support_level support_level;
};
```

```
#define AST_MODULE_INFO(keystr, flags_to_set, desc, fields...)		\
	static struct ast_module_info	__mod_info = {			\
		.name = AST_MODULE,					\
		.flags = flags_to_set,					\
		.description = desc,					\
		.key = keystr,						\
		.buildopt_sum = AST_BUILDOPT_SUM,			\
		fields							\
	};								\
	static void  __attribute__((constructor)) __reg_module(void) {	\
		ast_module_register(&__mod_info);			\
	}								\
	static void  __attribute__((destructor)) __unreg_module(void) {	\
		ast_module_unregister(&__mod_info);			\
	}								\
	struct ast_module *AST_MODULE_SELF_SYM(void)  {			\
		return __mod_info.self;					\
	}								\
	static const struct ast_module_info *ast_module_info = &__mod_info
```

Подключение модуля:

```
AST_MODULE_INFO(ASTERISK_GPL_KEY, AST_MODFLAG_DEFAULT, "Skeleton (sample) Application",
	.support_level = AST_MODULE_SUPPORT_CORE,
	.load = load_module,
	.unload = unload_module,
	.reload = reload_module,
);
```

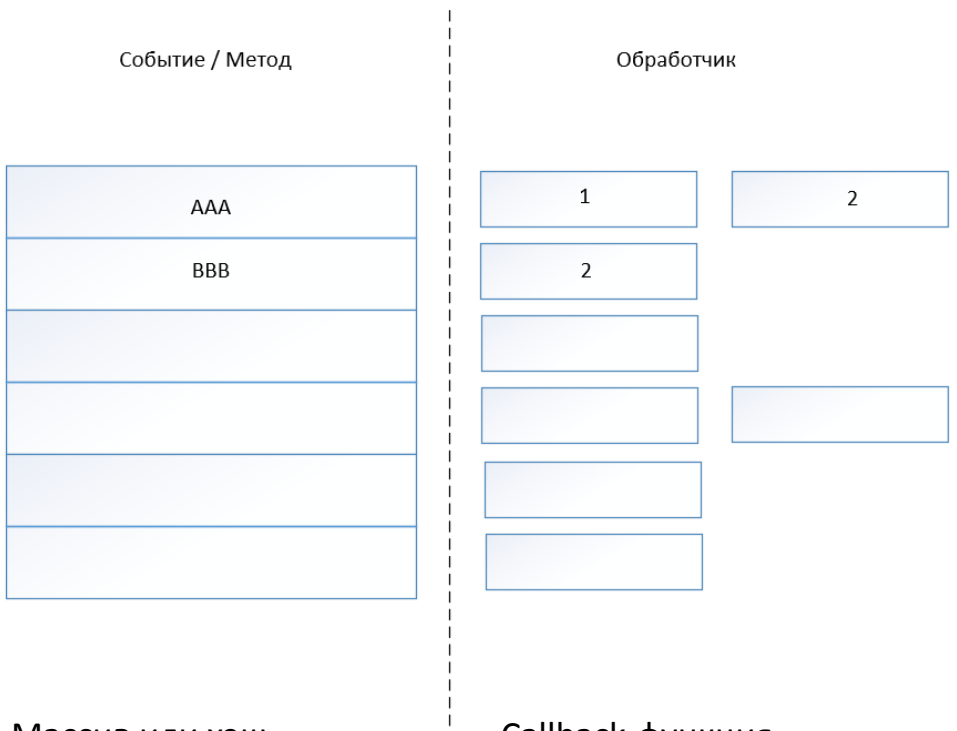
02

Шина

Работа шины

- Зарегистрировать событие, повесить обработчик
- Запись в шину (генерация события)
- Чтение из шины

Гарантия доставки сообщения зависит от реализации - чаще гарантии нет. Т.е. если обработчик не зарегистрирован, сообщение будет потеряно

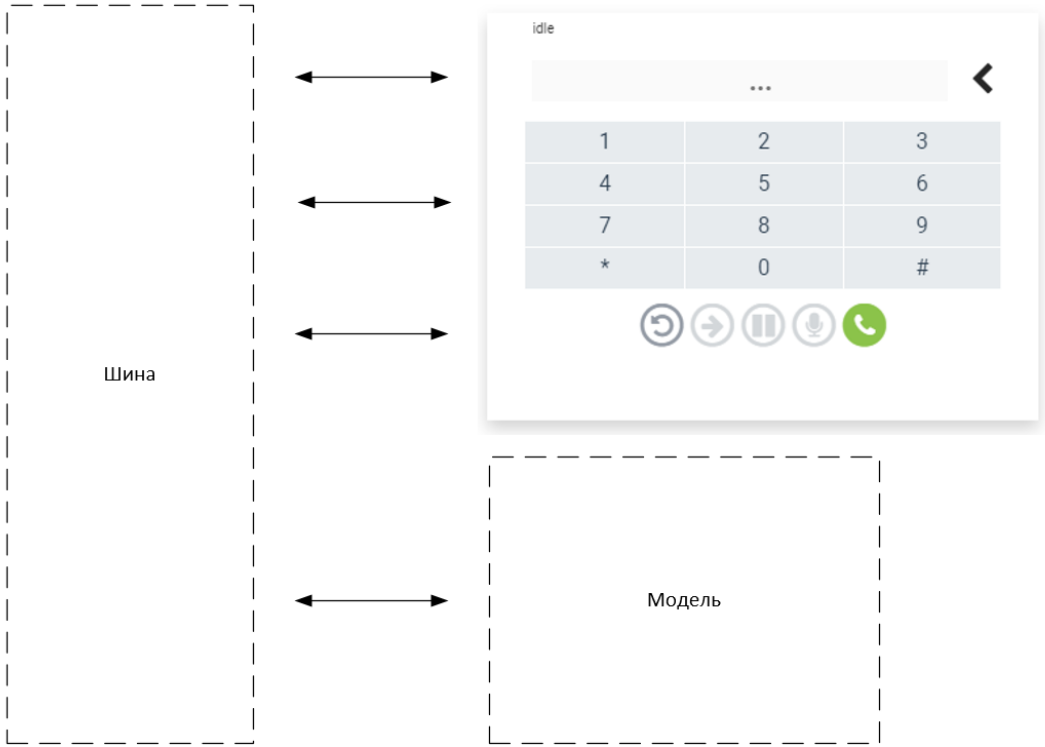


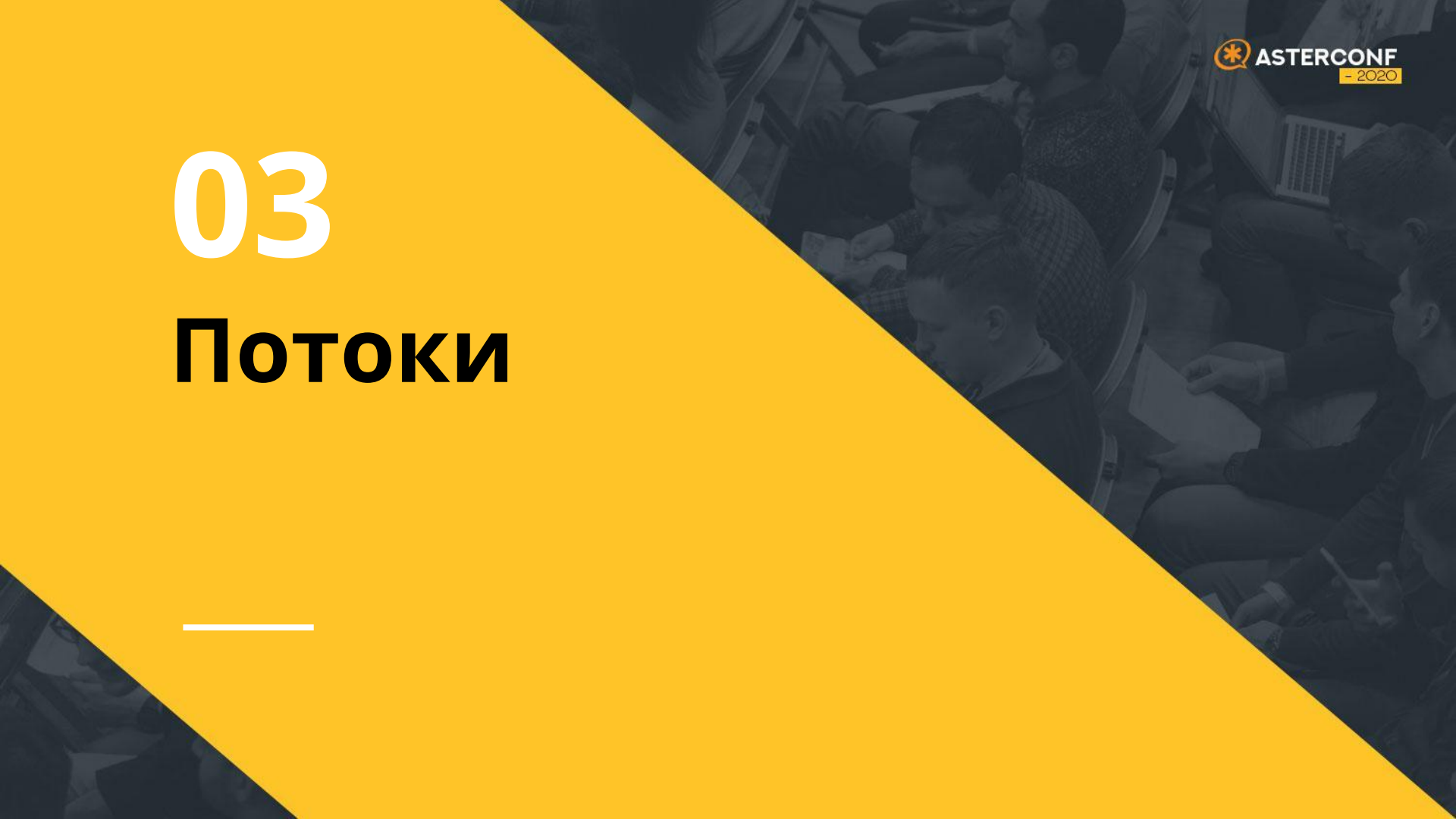
Массив или хэш-таблица

Callback-функция

Взаимодействие модулей

- register_event(event, handler)
- ssb_fire(event, data)





03

Потоки

Sonic

Sonic - это животное, которое за одну секунду может выпить чашечку кофе, принять ванну, сходить в кино и немного поспать

Задача

Что будет, если в одной комнате закрыть четыре соника?

Что если попросить их, чтобы они записывали свои действия в журнал?

Синхронизация

Журнал - это разделяемый объект.

Механизмом синхронизации будет ручка, которую Соник должен будет взять, прежде чем писать.

mutex

pthread_mutex_lock()

pthread_mutex_trylock()

pthread_mutex_unlock()

spin

pthread_spin_lock()

pthread_spin_trylock()

pthread_spin_unlock()

Возможные проблемы синхронизации

- Взаимная блокировка (deadlock)
- Блокировок слишком много
- Блокировок недостаточно

Как заблокировать два канала?

НЕ ДЕЛАЙ ТАК

```
ast_channel_lock(chan1);
ast_channel_lock(chan2);
```

Правильная блокировка двух каналов

```
#define ast_channel_lock_both(chan1, chan2) do {                \
        ast_channel_lock(chan1);                                \
        while (ast_channel_trylock(chan2)) {                    \
                ast_channel_unlock(chan1);                      \
                sched_yield();                                  \
                ast_channel_lock(chan1);                        \
        }                                                       \
} while (0)
```

СПАСИБО
ЗА ВНИМАНИЕ!

Штейнлихт Олег

iptelefon.su

ASTERCONF
- 2020