Cellular Network Infrastructure - Bug #4865

some osmo projects don't generate coredump upon SIGABRT

11/20/2020 05:18 PM - pespin

Status:	Resolved	**Start date:**	11/20/2020
Priority:	Normal	**Due date:**	
Assignee:	pespin	**% Done:**	100%
Category:			
Target version:			
Spec Reference:			

Description

See for instance osmo-bsc:

```
static void signal_handler(int signal)
{
    fprintf(stdout, "signal %u received\n", signal);

    switch (signal) {
    case SIGINT:
    case SIGTERM:
        bsc_shutdown_net(bsc_gsmnet);
        osmo_signal_dispatch(SS_L_GLOBAL, S_L_GLOBAL_SHUTDOWN, NULL);
        sleep(3);
        exit(0);
        break;
    case SIGABRT:
        /* in case of abort, we want to obtain a talloc report
         * and then return to the caller, who will abort the process */
    case SIGUSR1:
        talloc_report(tall_vty_ctx, stderr);
        talloc_report_full(tall_bsc_ctx, stderr);
        break;
    default:
        break;
    }
}

    signal(SIGINT, &signal_handler);
    signal(SIGTERM, &signal_handler);
    signal(SIGABRT, &signal_handler);
    signal(SIGUSR1, &signal_handler);
    signal(SIGUSR2, &signal_handler);
```

So when a process sends SIGABRT to it (killall -ABRT osmo-bsc), it will simply print talloc contexts and continue executing normally (and no coredump dumped). That's definetly not what we want here. Upon SIGABRT, we want the process to produce a coredump and exit.

In order to do so, after calling talloc_report we have to call the default SIGABRT handler.

That can be done by keeping reference to the older signal handler at startup when we voerride the signal:

```
sighandler_t default_sigabrt; /* global varialbe */
default_sigabrt = signal(SIGABRT, &signal_handler);
```

Then in our signal_handler:

```
static void signal_handler(int signal)
{
...
```

```
    case SIGABRT:
        talloc_report(tall_vty_ctx, stderr);
        talloc_report_full(tall_bsc_ctx, stderr);
        default_sigabrt(signal);
        break;
...
}
```

History

#1 - 11/20/2020 07:50 PM - laforge

Nice catch!

#2 - 11/25/2020 03:57 PM - pespin

So this issue only appears when sending plain SIGABRT, abort() should handle this gracefully. From "man abort":

```
DESCRIPTION
       The  abort()  function first unblocks the SIGABRT signal, and then raises that signal for the calling p
rocess (as though raise(3) was called).  This
       results in the abnormal termination of the process unless the SIGABRT signal is caught and the signal h
andler does not return (see longjmp(3)).

       If the SIGABRT signal is ignored, or caught by a handler that returns, the abort() function will still
terminate  the  process.   It  does  this  by
       restoring the default disposition for SIGABRT and then raising the signal for a second time.
```

So abort takes care of calling SIG_DFL if our handler returned. But in the event a plain SIGBART is sent, the program will continue without generating a coredump (because the code generating the coredump is inside SIGABRT SIG_DFL).

So the best is probably to do this in the SIGABRT signal handler:

```
signal(SIGABRT, SIG_DFL);
raise (SIGABRT);
```

#3 - 11/25/2020 04:11 PM - pespin

Fixed (tested) here:
https://gerrit.osmocom.org/c/osmo-bsc/+/21336 main: generate coredump and exit upon SIGABRT received

I will now look at similar issue in other osmocom projects.

#4 - 11/25/2020 06:25 PM - pespin

- Status changed from New to Feedback

- % Done changed from 0 to 90

I submitted a bunch of patches to gerrit fixing same issue in the other osmocom projects.
Once merged ticket can be closed.

#5 - 12/09/2020 01:19 PM - pespin

- Status changed from Feedback to Resolved

- % Done changed from 90 to 100

Merged, closing.