CU CS 4995-03: Internet Systems Programming

Process Environment

Overview

- `main`
- process termination
- command-line arguments
- environment
- limits: `getrlimit` and `setrlimit`

main

```
extern char **environ;
int main(int argc, char *argv[], char *envp[]);
{
  int i;

  for (i = 0; i < argc; i++) {
    printf("argv[%d] = %s\n", i, argv[i]);
  }
  for (i = 0; envp[i]; i++) {
    printf("env[%d] = %s\n", i, envp[i]);
  }
  return 0;
}
```

`envp` usually omitted.

Process Termination

- return from `main`
- call `exit(int)`
- call `_exit(int)`
- call `abort(void)`
- terminate by signal

Conceptually: `exit(main(argc,argv,envp));`

exit and _exit

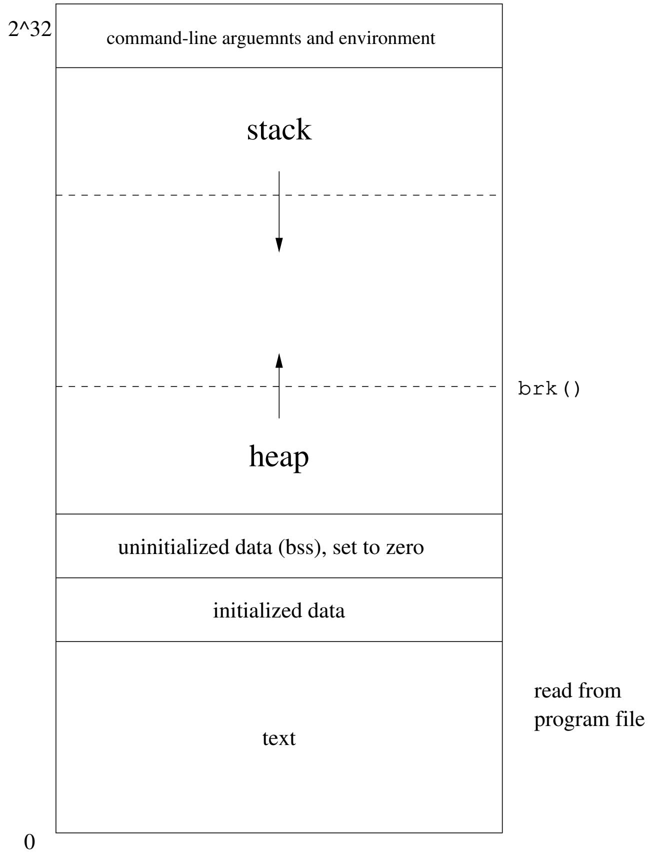
- `exit`: close sockets (`fclose()`); why?
- call `atexit()`
- `exit_`: return to kernel
- exit status: 0 ok, anything else bad
- e.g., csh: `echo $status`

atexit

```
void handler(void)
{
  printf("going...\n");
}
atexit(handler);
```

- reverse order of registration

Memory Layout




```
$ size main
text   data   bss    dec    hex filename
1687    228    40   1955    7a3 main
```

Process Limits

```
struct rlimit rl;
getrlimit(RLIMIT_CORE, &rl);
rl.rlim_cur = 0;
setrlimit(RLIMIT_CORE, &rl);
```

`RLIMIT_CORE`	core file
`RLIMIT_CPU`	CPU time
`RLIMIT_FSIZE`	file size

generates signals when exceeding limits
use `ulimit` in shell