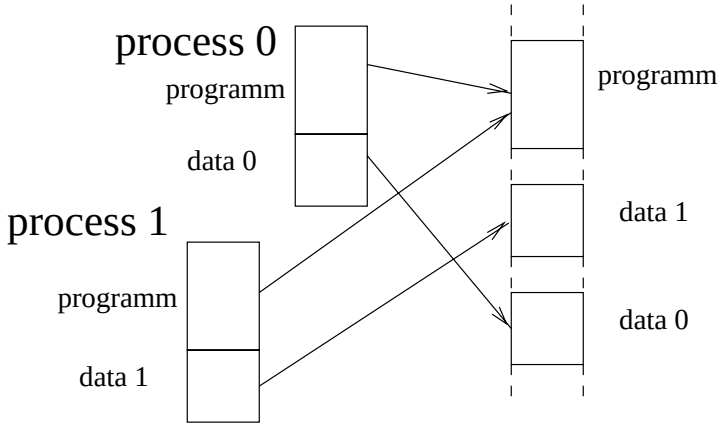
Processes

- Process creation
- C functions
- Examples

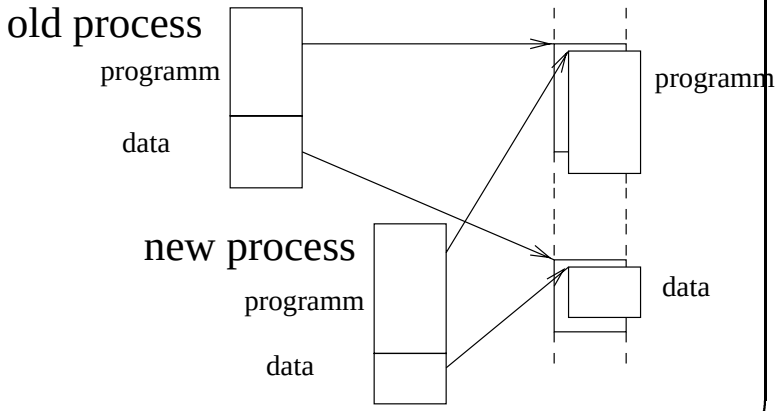
Process creation

```
$pid=fork();
if ($pid == 0)
    { ........ child .........  }
elsif ($pid > 0)
    { ........ parent ......... }
else
    { ........ error .........  }
```


Process memory



Function – exec



Signals

```
signal(SIGQUIT,SIG_DFL);     /* reset to default handling */
signal(SIGQUIT,SIG_IGN);     /* ignore signal */
signal(SIGQUIT,proc);        /* set custom handler */
kill(pid,SIGQUIT);           /* send signal */
void proc( int sig ) { ... } /* example signal handler */
```


SIGHUP 1 – hangup
SIGINT 2 – interrupt (rubout)
SIGQUIT 3 – quit (ASCII FS)
SIGILL 4 – illegal instruction (not reset when caught)
SIGTRAP 5 – trace trap (not reset when caught)
SIGIOT 6 – IOT instruction
SIGABRT 6 – used by abort, replace SIGIOT in the future
SIGEMT 7 – EMT instruction
SIGFPE 8 – floating point exception
SIGKILL 9 – kill (cannot be caught or ignored)
SIGBUS 10 – bus error
SIGSEGV 11 – segmentation violation
SIGSYS 12 – bad argument to system call
SIGPIPE 13 – write on a pipe with no one to read it
SIGALRM 14 – alarm clock
SIGTERM 15 – software termination signal from kill
SIGUSR1 16 – user defined signal 1

Waiting for a signal

```
int pause(void);
```

pause() suspends the calling process until it receives a signal. The signal must be one that is not currently set to be ignored by the calling process.

Waiting for a child

```
pid_t waitpid(pid_t pid, int *stat_loc, int options);
```

pid – ID of the process
stat_loc – where to store the exit status
options – options:
- **WCONTINUED** – report also continued processes
- **WNOHANG** – do not suspend execution of the calling process
- **WNOWAIT** – we wish to wait again for the same process
- **WUNTRACED** – report also stopped processes