Humboldt University

Computer Science Department
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Operating Systems Principles

Lab 2 – Command Shell

Zweites Praktikum

- 2 Wochen Zeit zum Lösen der Aufgaben
  - Aufgabenstellung auf der SAR Website
  - Abgabe über GOYA
- In dieser Woche
  - Fragen zur aktuellen Aufgabe
  - Erläutern neuen Aufgabenstellung
  - System Architecture & Privilege Levels
  - (Kernel debugging)
- Nächsten 2 Wochen
  - Zeit zur Bearbeitung
  - Konsultation
- Nächstes Treffen: **18.05.2012**

Lab 2 explained

Gesucht: eine einfache Shell

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Lab 2 explained

Gesucht: eine einfache Shell

1. Bereitstellen einer **Kommandozeile** und **Ausführen eines Programms** mit Kommandozeilenparametern
   1. Aktuelles Verzeichnis relativ zum Einstiegspunkt als Prompt

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Lab 2 explained

Gesucht: eine einfache Shell

1. Bereitstellen einer **Kommandozeile** und **Ausführen eines Programms** mit Kommandozeilenparametern
   1. Aktuelles Verzeichnis relativ zum Einstiegspunkt als Prompt
2. Ausführen eines Programms im **Hintergrund** mit **Rückgabe der PID**

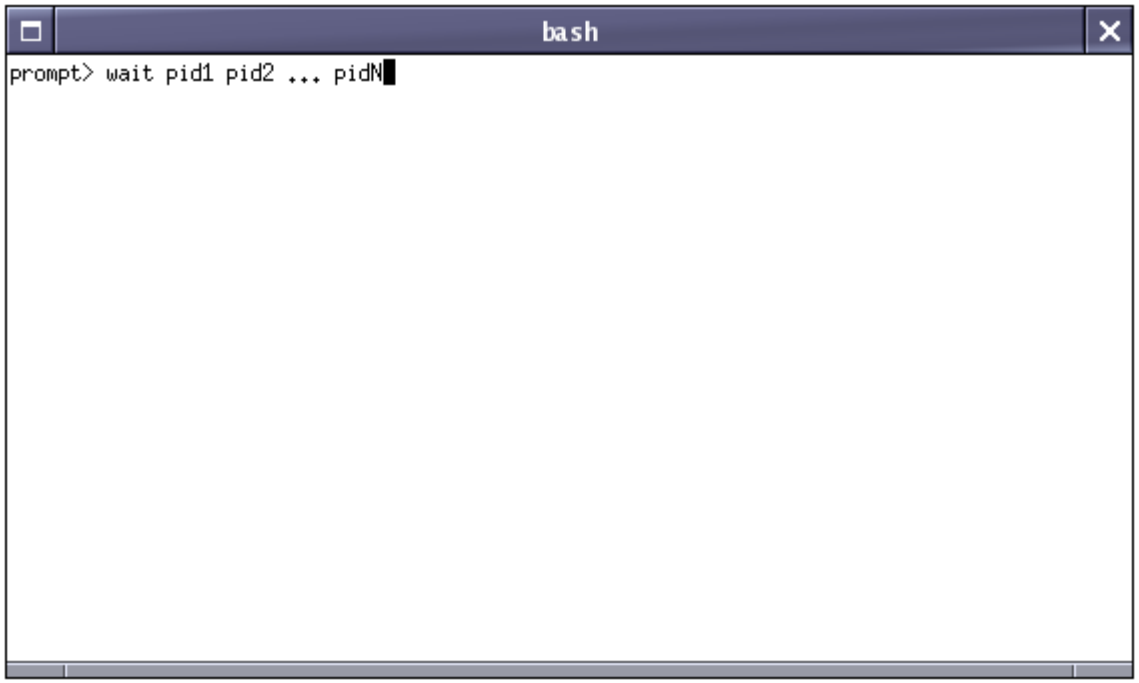
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Lab 2 explained

Gesucht: eine einfache Shell

1. Bereitstellen einer **Kommandozeile** und **Ausführen eines Programms** mit Kommandozeilenparametern
   1. Aktuelles Verzeichnis relativ zum Einstiegspunkt als Prompt
2. Ausführen eines Programms im **Hintergrund** mit **Rückgabe der PID**
3. Wechseln des **Arbeitsverzeichnisses**

Lab 2 explained



Gesucht: eine einfache

1. Bereitstellen einer **Kom** **Programms** mit Komm
   1. Aktuelles Verzeichnis
2. Ausführen eines Progra
3. Wechseln des **Arbeitsv**
4. **Warten** auf Prozesse.
   a) Warten auf eine **Teilmenge** der aktuell ausgeführten Hintergrundprozesse pid1 ... pidN
   b) Das "wait" soll mit Ctrl+C **unterbrechbar** sein
   c) Bei Terminierung: möglichst viel Informationen über den **Endzustand** des Prozesses ausgeben
      - Grund des Terminierens, Rückgabewert, …

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

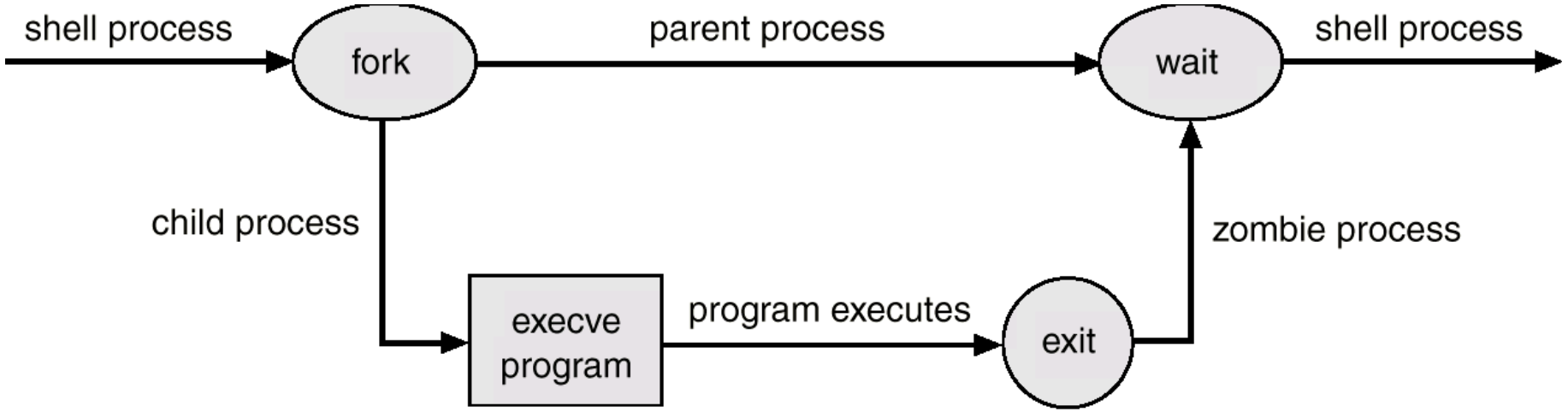
Lab 2 explained

Gesucht: eine einfache Shell

1. Bereitstellen einer **Kommandozeile** und **Ausführen eines Programms** mit Kommandozeilenparametern
   1. Aktuelles Verzeichnis relativ zum Einstiegspunkt als Prompt
2. Ausführen eines Programms im **Hintergrund** mit **Rückgabe der PID**
3. Wechseln des **Arbeitsverzeichnisses** („cd dir“)
4. **Warten** auf Prozesse.
   a) Warten auf eine **Teilmenge** der aktuell ausgeführten Hintergrundprozesse pid1 ... pidN
   b) Das "wait" soll mit Ctrl+C **unterbrechbar** sein
   c) Bei Terminierung: möglichst viel Informationen über den **Endzustand** des Prozesses ausgeben
      - Grund des Terminierens, Rückgabewert, …
5. **Beenden der Shell** mit „exit“

Humboldt Universität Informatik
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Command Line Shell



Systems Architecture Group
http://sar.informatik.hu-berlin.de

fork

- **NAME**
  - fork
- **SYNOPSIS**
  - #include <unistd.h>
    pid_t fork(void);
- **DESCRIPTION**
  - Creates a new process
  - The child process is an exact copy of the parent process except …

http://www.opengroup.org/onlinepubs/009695399/functions/fork.html

exec*

- **NAME**
  - exec* - execl, execv, execle, execve, execlp, execvp

- **SYNOPSIS**
  - #include <unistd.h>

```
int execl(const char *path, const char *arg0, ... /*, (char *)0 */);
int execv(const char *path, char *const argv[]);
int execle(const char *path, const char *arg0, ... /*,
      (char *)0, char *const envp[]*/);
int execve(const char *path, char *const argv[], char *const envp[]);
int execlp(const char *file, const char *arg0, ... /*, (char *)0 */);
int execvp(const char *file, char *const argv[]);
```

- **DESCRIPTION**
  - The *exec* family of functions shall replace the current process image with a new process image

http://www.opengroup.org/onlinepubs/009695399/functions/exec.html

exit

- **NAME**
  - exit

- **SYNOPSIS**
  - #include <stdlib.h>

    void exit(int *status*);

- **DESCRIPTION**
  - Terminate a process

http://www.opengroup.org/onlinepubs/009695399/functions/exit.html

wait

- **NAME**
  - wait, waitpid
- **SYNOPSIS**
  - #include <sys/wait.h>

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

- **DESCRIPTION**
  - Wait for a child process to stop or terminate
  - Obtain status information pertaining to one of the caller's child processes
  - If status information is available for two or more child processes, the order in which their status is reported is unspecified.

http://www.opengroup.org/onlinepubs/009695399/functions/wait.html

signal

- **NAME**
  - signal

- **SYNOPSIS**
  - #include <signal.h>

    void (*signal(int *sig*, void (**func*)(int)))(int);

- **DESCRIPTION**
  - signal management
  - Determines how to handle the receiption of the signal number *sig* is to be subsequently handled
    - SIG_DFL - default handling for that signal
    - SIG_IGN - the signal shall be ignored
    - Otherwise, *func* points to a function ("signal handler") to be called when that signal occurs

http://www.opengroup.org/onlinepubs/009695399/functions/signal.html

kill

- **NAME**
  - kill

- **SYNOPSIS**
  - #include <signal.h>

    int kill(pid_t pid, int sig);

- **DESCRIPTION**
  - Sends signal to a process or a group of processes specified by *pid*
    - Pids <= 0 used to send a signal to more than one process
  - Return value: 0 if successful ; -1 otherwise
  - Signals (examples):
    - SIGSTOP, SIGCONT
    - SIGTERM, SIGKILL

http://pubs.opengroup.org/onlinepubs/7908799/xsh/kill.html

chdir

- **NAME**
  - chdir
- **SYNOPSIS**
  - #include <unistd.h>

    int chdir(const char *_path_);
- **DESCRIPTION**
  - Changes working directory

http://www.opengroup.org/onlinepubs/009695399/functions/chdir.html

getcwd

- **NAME**
  - getcwd
- **SYNOPSIS**
  - #include <unistd.h>

    char *getcwd(char **buf*, size_t *size*);
- **DESCRIPTION**
  - Get the pathname of the current working directory

http://www.opengroup.org/onlinepubs/009695399/functions/getcwd.html