[CSED211] Introduction to Computer Software Systems

Lab 7: Shell Lab

Sucheol Lee

COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.11.28

Today's Agenda

- Background
  - Shell
  - Signal

- Shell Lab

Shell Programs

- Shell: An interactive command-line interpreter that runs programs on behalf of user
  - Command line: A sequence of ASCII text words

- Common examples: Bash (Bourne-again shell)
  - Linux default

- Most applications in Linux (command line) are run through shell

Basic Functions of Shell

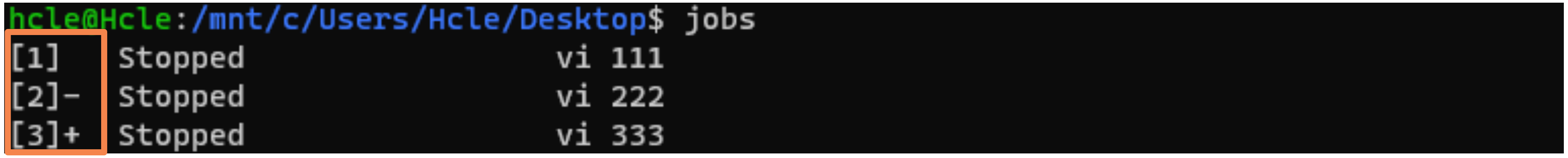
- **`jobs`**
  - Lists the stopped and running background job

- **`fg <job_id>`**
  - Makes a stopped or running background job run at the foreground

- **`bg <job_id>`**
  - Makes a stopped background job run at the background

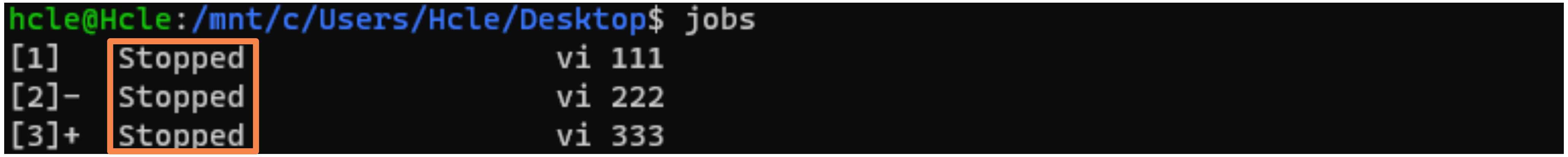
Basic Functions of Shell (Cont.)

- `jobs` command example



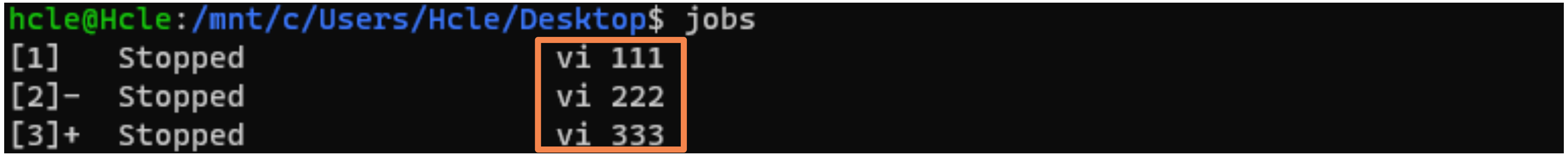
```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]   Stopped                 vi 111
[2]-  Stopped                 vi 222
[3]+  Stopped                 vi 333
```

Job id, spec



```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]   Stopped                 vi 111
[2]-  Stopped                 vi 222
[3]+  Stopped                 vi 333
```

Status

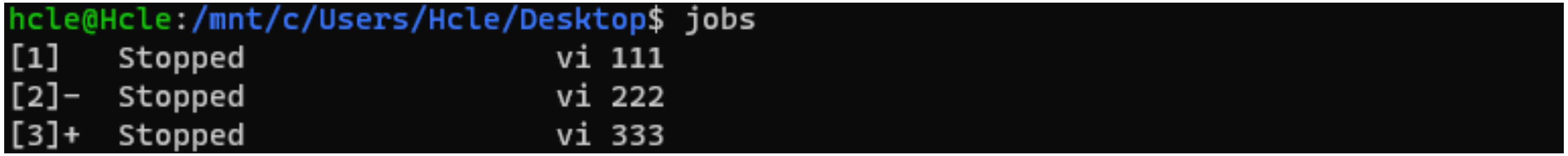


```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]   Stopped                 vi 111
[2]-  Stopped                 vi 222
[3]+  Stopped                 vi 333
```

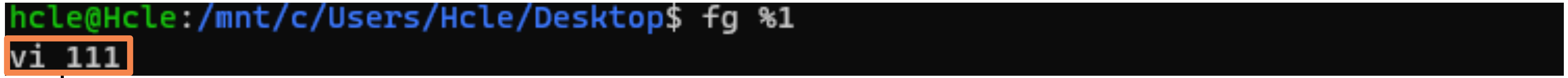
Job name

Basic Functions of Shell (Cont.)

- `fg` command example



```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]   Stopped                 vi 111
[2]-  Stopped                 vi 222
[3]+  Stopped                 vi 333
```

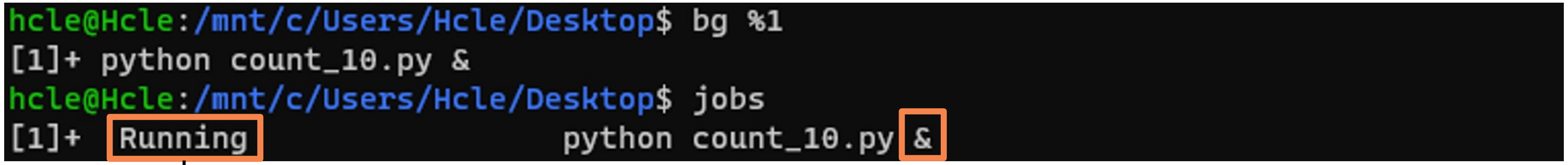


```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ fg %1
vi 111
```

→ **`vi 111`**: A stopped background job is now running in the foreground

Basic Functions of Shell (Cont.)

- **bg** command example



```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ bg %1
[1]+ python count_10.py &
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]+  Running                 python count_10.py &
```

A stopped background job is now running in the background

Signal

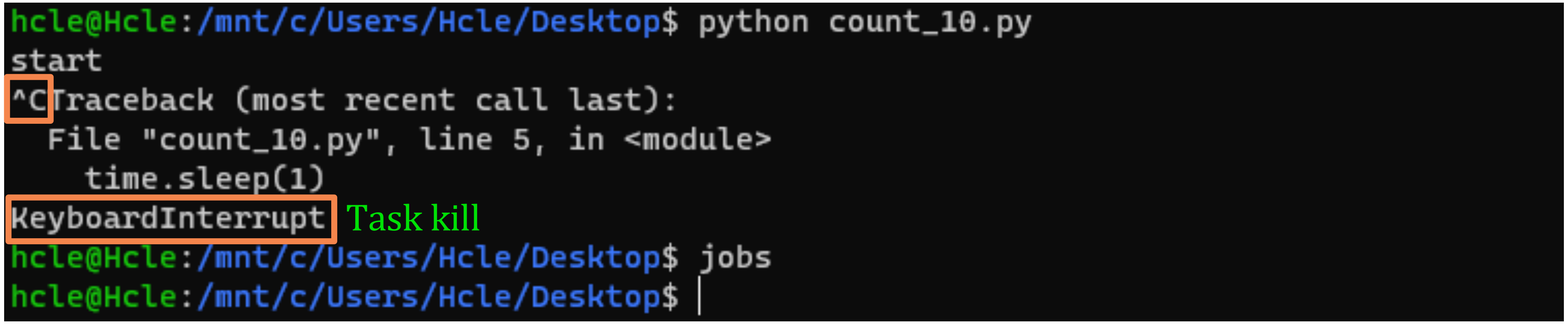
- A small message that notifies a process of system event
  - Sent from the kernel to a process
    - Sometimes at the request from another process
  - Signal ID: Small integer ID (1~30) that represents signal type

ID	Name	Default Action	Corresponding Event
2	`SIGINT`	Terminate	Interrupt (e.g., **`ctl-c`** from keyboard)
9	`SIGKILL`	Terminate	Kill process (cannot override or ignore)
11	`SIGSEGV`	Terminate & Dump	Segmentation violation
14	`SIGALRM`	Terminate	Timer signal
17	`SIGCHLD`	Ignore	Child stopped or terminated
19	`SIGSTOP`	Stop	Stop process (cannot override or ignore)

Signal Handling

- Kernel makes the destination process react to the delivery of the signal

- e.g., `SIGINT` signal
  - `CTRL-C` sends a `SIGINT` to every job in the foreground process group
  - Default action: Terminate each process

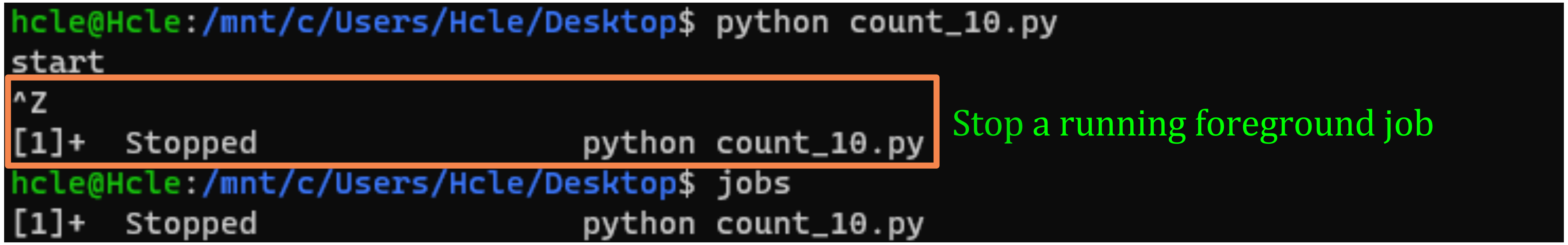


```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ python count_10.py
start
^CTraceback (most recent call last):
  File "count_10.py", line 5, in <module>
    time.sleep(1)
KeyboardInterrupt
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$
```

Signal Handling (Cont.)

- Kernel makes the destination process react to the delivery of the signal

- e.g., `SIGSTP` signal
  - `CTRL-Z` sends a `SIGTSTP` to every job in the foreground process group
  - Default action: Suspend each process
    - To run task again, use `fg %{job_id}` or `bg %{job_id}`



```
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ python count_10.py
start
^Z
[1]+  Stopped                 python count_10.py
hcle@Hcle:/mnt/c/Users/Hcle/Desktop$ jobs
[1]+  Stopped                 python count_10.py
```

Signal Set and Mask

- Signal set
  - Data type that lets a process keep track of groups of signals (`sigset_t`)
    - `sigemptyset`: Initializes the signal set set to exclude all of the defined signals
    - `sigaddset`: Adds the signal signum to the signal set
      - It modifies the signal set, but does not block or unblock any signals

- Signal mask
  - Collection of signals that are currently blocked
    - Each process has its own signal mask
      - When you create a new process, it inherits its parent's mask
    - `sigprocmask`: Modifies the current signal mask of a process with the signal set
      - It takes a signal set as input to add or remove signals from the current mask

Today's Agenda

- Background
  - Shell
  - Signal

- Shell Lab

Shell Lab: Overview

- Write a simple shell (`tsh`) that supports the following functionalities
  1. Running command
  2. Launching foreground job
  3. Launching background job
  4. Switching between foreground and background job

- Complete seven below functions to support the above functionalities
  - `eval`, `builtin_cmd`, `do_bgfg`, and `waitfg`
  - `SIGCHLD`, `SIGINT`, and `SIGSTP` handler

- Helper function is provided in the source file

Shell Lab: Overview (Cont.)

- Due: 12/11 23:59 (Late submission will not be accepted)

- Submit a code file and your lab report (in pdf)
  - Source code name: `[student id].c` (e.g., `20231234.c`)
    - Complete `tsh.c` file and rename the code file to [student id].c
  - Report name: `[student id].pdf` (e.g., `20231234.pdf`)

Shell Lab: Running Command

1. Runs a command
    - Built-in commands (`quit`, `jobs`, `bg`, and `fg`)
        - If the user has typed a built-in command, then executes it immediately without forking child process
            - e.g., `tsh> jobs`
                - Prints list of jobs including both running and stopped jobs
            - e.g., `tsh> quit`
                - Quits `tsh` and return to bash
    - Other commands
        - Forks child process and manages running application as child process
            - e.g., `tsh> /bin/ls -l -h`
                - Executes "`/bin/ls`" with arguments "`-l`" and "`-h`"
                - `argv[0] = "/bin/ls", argv[1] = "-l", argv[2]="-h"`

Shell Lab: Launching Foreground/Background Job

2. Foreground job launching
   - Runs the command in foreground and waits for its ending
     - e.g., `tsh> /bin/ls -l -h`
       - Shell executes "`/bin/ls`" with "`-l -h`"
       - Waits for it to finish before other application runs
   - Every application run is foreground by default

3. Background job launching
   - Runs application in background
     - `tsh` can run many background jobs
   - `&`: Added to end of command
     - e.g., `tsh>./myprogram &`
       - Specifies this needs to be run in the background

Shell Lab: Switch Between Foreground/Background

4. Foreground/background management
   - Moves jobs between foreground and background or changes the job status
     - e.g., `tsh> fg <job_id>`
       - Changes a stopped or running background job to a running foreground job
     - e.g., `tsh> bg <job_id>`
       - Changes a stopped background job to a running background job

Shell Lab: Functions to be Implemented

- **`eval(char *cmdline)`**
  - Evaluates the command line that the user has typed in
    - For built-in commands (**`quit`**, **`jobs`**, **`bg`** and **`fg`**), executes it on **`builtin_cmd`** function
    - Otherwise, forks a child process and runs the job in the context of the child
- **`builtin_cmd(char **argv)`**
  - Executes a built-in command immediately if the input command is one of built-in commands (**`quit`**, **`jobs`**, **`bg`**, and **`fg`**)
- **`do_bgfg(char **argv)`**
  - Executes the built-in **`bg`** and **`fg`** commands
- **`waitfg(pid_t pid)`**
  - Waits for a specified foreground job to complete

Shell Lab: Functions to be Implemented (Cont.)

- **`SIGCHLD handler`**
  - Reaps all available zombie children, but doesn't wait for any other currently running children to terminate
- **`SIGINT handler`**
  - Catches **`SIGINT`** from the kernel and sends it along to the foreground job
- **`SIGSTP handler`**
  - Catches **`SIGSTP`** from the kernel and suspends the foreground job by sending it a **`SIGTSTP`**

Shell Lab: Seven Functions to be Implemented (Cont.)

- `SIGCHLD handler`
  - Reaps all available zombie children, but doesn't wait for any other currently running children to terminate
- `SIGINT handler`
  - Catches `SIGINT` from the kernel and sends it along to the foreground job
- `SIGSTP handler`
  - Catches `SIGSTP` from the kernel and suspends the foreground job by sending it a `SIGTSTP`

Complete the functions within the provided `tsh.c` skeleton code

Shell Lab: Seven Functions to be Implemented (Cont.)

- Many helper functions provided
  - `parseline`: Parses the command line and build the argv array
  - `addjob` and `deletejob`: Adds or deletes the job
  - `clearjob`: Clears the entries in a job struct
  - `fgpid`: Returns PID of current foreground job
  - `getjobpid`: Finds a job on the job list by PID
  - `getjobid`: Finds a job on the job list by JID
- Four executable programs run as input commands in `tsh`
  - `myint`
  - `myspin`
  - `mysplit`
  - `mystop`

Shell Lab: Evaluation

- Score evaluation: Quiz (10%) + Test cases (40%) + Report (50%)

- Use the provided 'reference `tsh`' binary and 16 traces
  - Run `tsh` with each trace and check whether the output matches `tshref.out`

- Testing
  - Run two commands as shown below
    1. `$ make test01`
    2. `$ make rtest01`
  - The number (01 in the above example) indicates the trace number
    - Modify the number to test with other traces
  - If the above two results match, then you get 2.5 points for each trace

Shell Lab: Report Guideline

- Attach the important parts of your code to your report
- Explain how you built `tsh`
- Report should not exceed 10 pages and use font Arial and font size 11pt
- Include all references you refer to solve shell lab assignment in your report

Shell Lab: Submission Guideline

- Due: 12/11 23:59 (Late submission will not be accepted)

- Submit a code file and your lab report (in pdf)
  - Source code name: `[student id].c` (e.g., `20231234.c`)
    - Complete `tsh.c` file and rename the code file to [student id].c
  - Report name: `[student id].pdf` (e.g., `20231234.pdf`)

Cheating Policy

- You can refer to
  - Shell lab writeup, lab slides, and lecture slides
  - Internet sources that do not include answers or code related to the cache lab

- You must not refer to
  - ChatGPT with direct query for answers or parts of a solution
  - Code and reports from seniors who have already taken this course
  - Blogs or github repositories that contain solution codes

[CSED211] Introduction to Computer Software Systems

Lab 7: Shell Lab

Sucheol Lee

CAOS
COMPUTER ARCHITECTURE &
OPERATING SYSTEMS LABORATORY

2024.11.28