# 3230shell – A simple shell program

2022-23 Programming #1

## Task

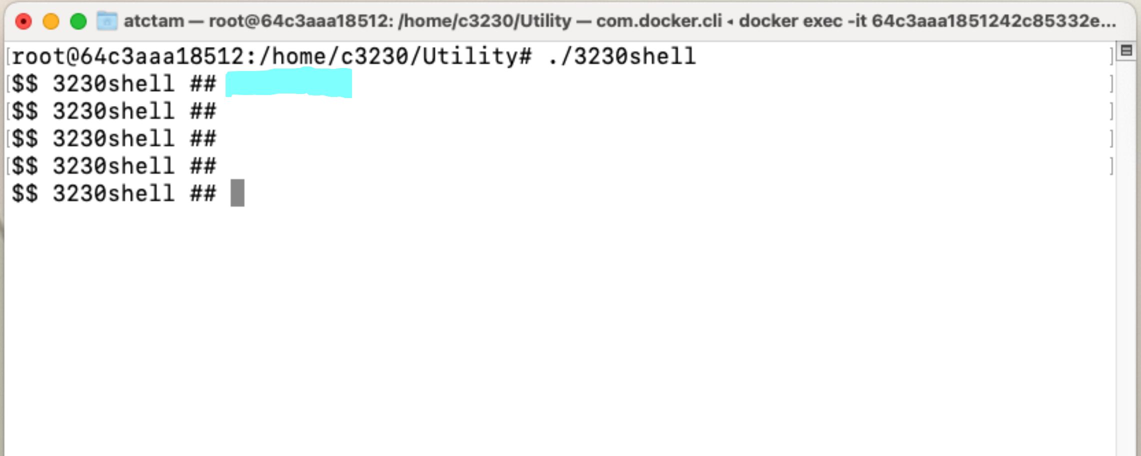
- You are going to implement a C program that acts as a simple interactive shell program, named as ***3230shell***.

## Objectives

- An assessment task related to ILO4 [Practicability]
  - Have hands-on practice in designing and developing a shell program, which involves the creation, management, and coordination of multiple processes
- A learning activity related to ILO 2a
- To learn how to use various important Unix system functions
  - to perform process creation and program execution
  - to support interaction between processes by using signals and pipes
  - to get process's running statistics

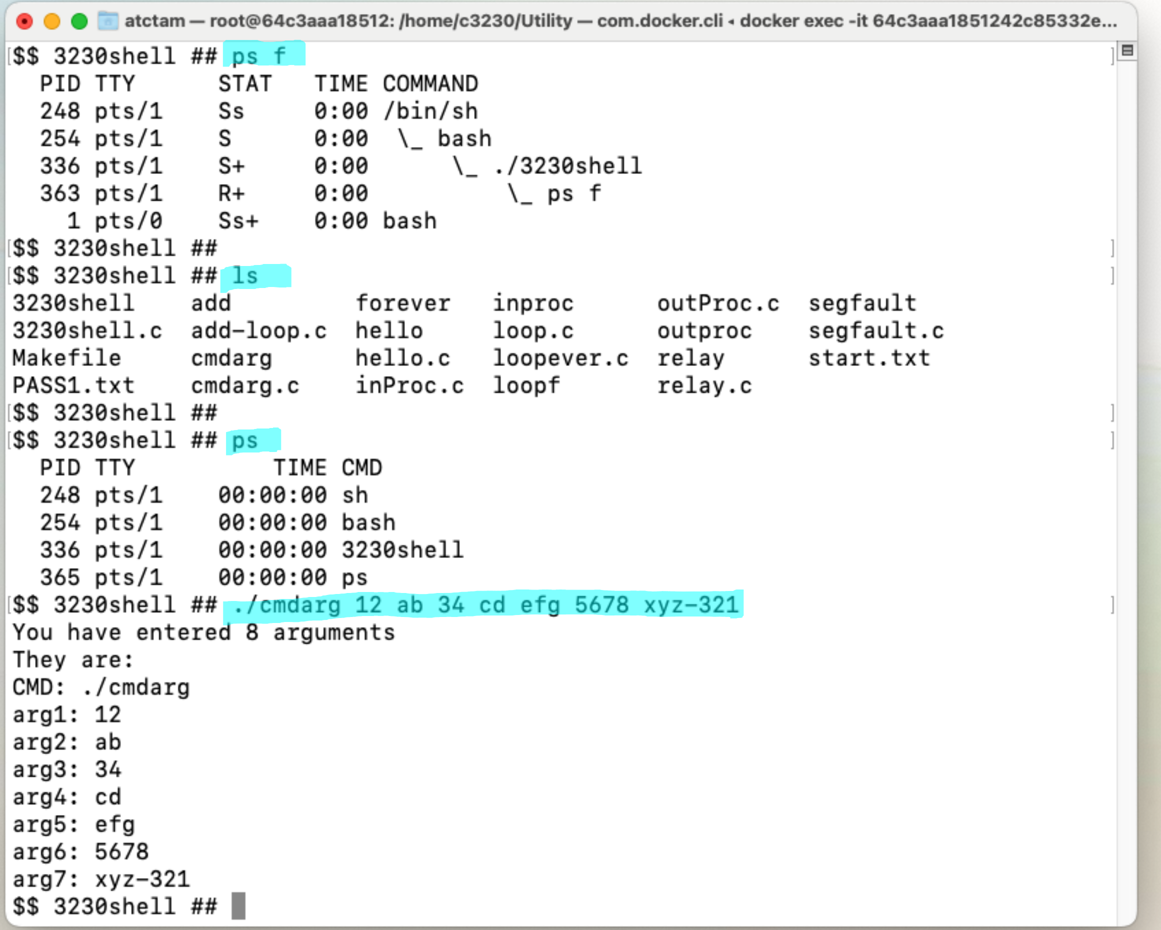
# Interactive Shell

- The 3230shell program accepts command(s) from user and executes the corresponding program(s) with the given argument list.
- Child runs in the foreground
  - 3230shell waits for the child to terminate



atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
root@64c3aaa18512:/home/c3230/Utility# ./3230shell
$$ 3230shell ##
$$ 3230shell ##
$$ 3230shell ##
$$ 3230shell ##
$$ 3230shell ##
```

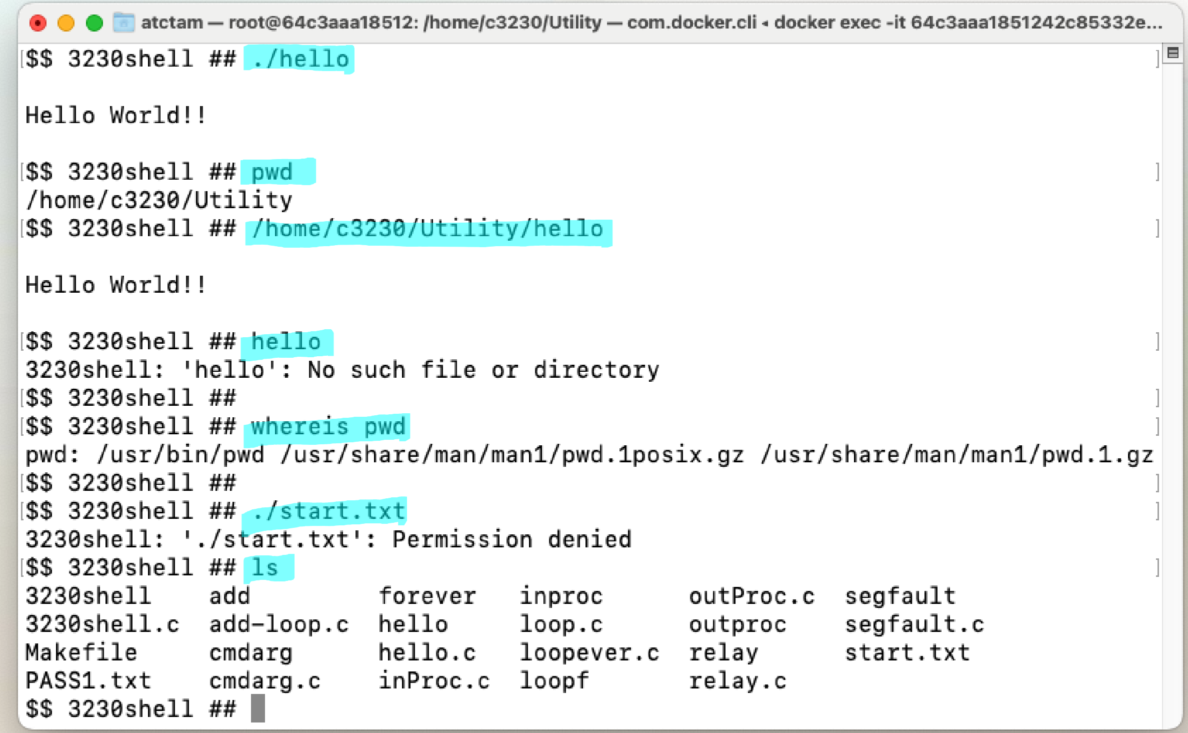


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## ps f
  PID TTY      STAT   TIME COMMAND
  248 pts/1    Ss     0:00 /bin/sh
  254 pts/1    S      0:00  \_ bash
  336 pts/1    S+     0:00      \_ ./3230shell
  363 pts/1    R+     0:00          \_ ps f
    1 pts/0    Ss+    0:00 bash
$$ 3230shell ##
$$ 3230shell ## ls
3230shell    add         forever    inproc      outProc.c  segfault
3230shell.c  add-loop.c  hello      loop.c      outproc    segfault.c
Makefile     cmdarg      hello.c    loopever.c  relay      start.txt
PASS1.txt    cmdarg.c    inProc.c   loopf       relay.c
$$ 3230shell ##
$$ 3230shell ## ps
  PID TTY          TIME CMD
  248 pts/1    00:00:00 sh
  254 pts/1    00:00:00 bash
  336 pts/1    00:00:00 3230shell
  365 pts/1    00:00:00 ps
$$ 3230shell ## ./cmdarg 12 ab 34 cd efg 5678 xyz-321
You have entered 8 arguments
They are:
CMD: ./cmdarg
arg1: 12
arg2: ab
arg3: 34
arg4: cd
arg5: efg
arg6: 5678
arg7: xyz-321
$$ 3230shell ##
```

# Feature – locate and execute any valid command

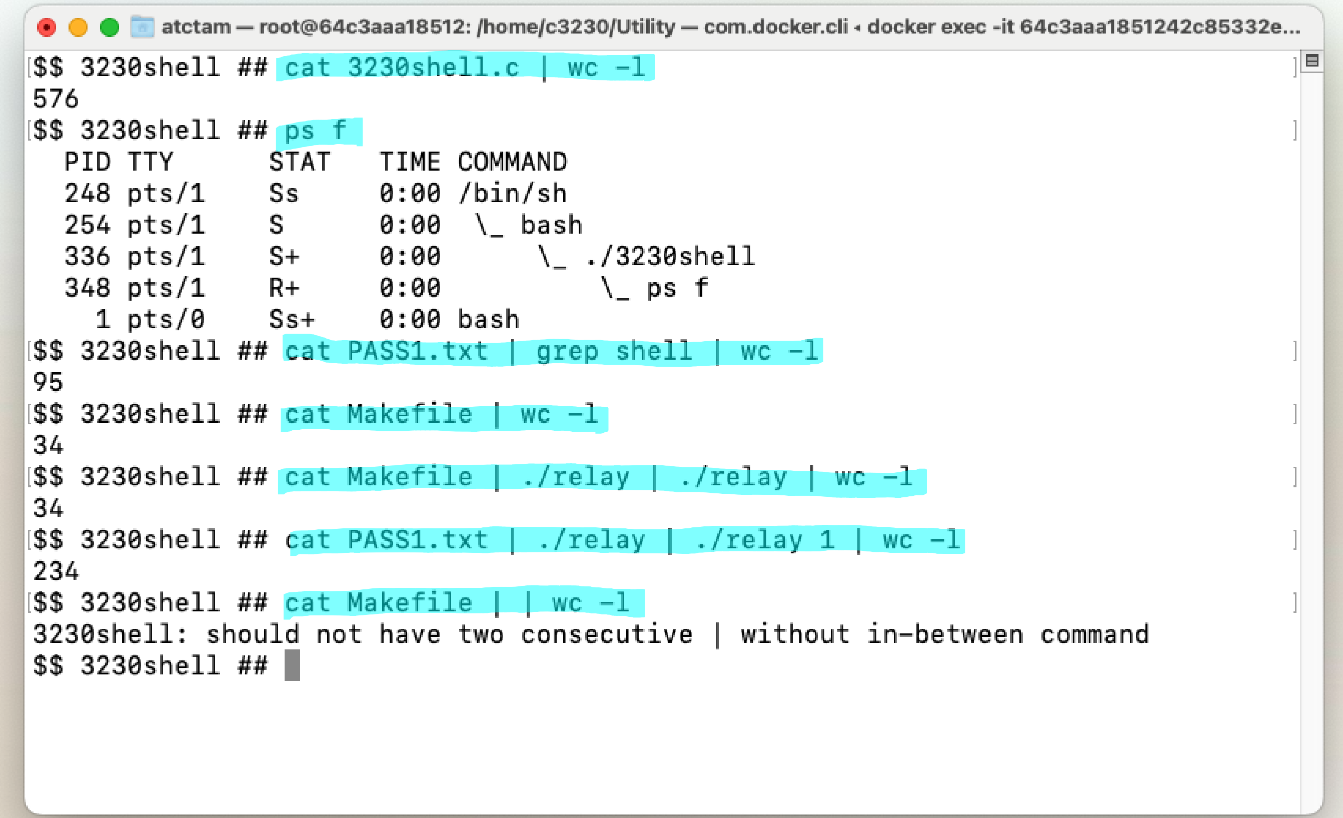
- It is able to locate and execute any valid program (i.e. programs that are already compiled) by giving an absolute path (starting with /) or a relative path (starting with ./) or by searching directories under the $PATH environment variable.



```
atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...
$$ 3230shell ## ./hello

Hello World!!

$$ 3230shell ## pwd
/home/c3230/Utility
$$ 3230shell ## /home/c3230/Utility/hello

Hello World!!

$$ 3230shell ## hello
3230shell: 'hello': No such file or directory
$$ 3230shell ##
$$ 3230shell ## whereis pwd
pwd: /usr/bin/pwd /usr/share/man/man1/pwd.1posix.gz /usr/share/man/man1/pwd.1.gz
$$ 3230shell ##
$$ 3230shell ## ./start.txt
3230shell: './start.txt': Permission denied
$$ 3230shell ## ls
3230shell     add          forever    inproc       outProc.c   segfault
3230shell.c   add-loop.c   hello      loop.c       outproc     segfault.c
Makefile      cmdarg       hello.c    loopever.c   relay       start.txt
PASS1.txt     cmdarg.c     inProc.c   loopf        relay.c
$$ 3230shell ##
```

## Feature - | pipe

- It supports the | (pipe) operator between commands.
- Assume no more than 4 pipes (i.e., 5 commands)

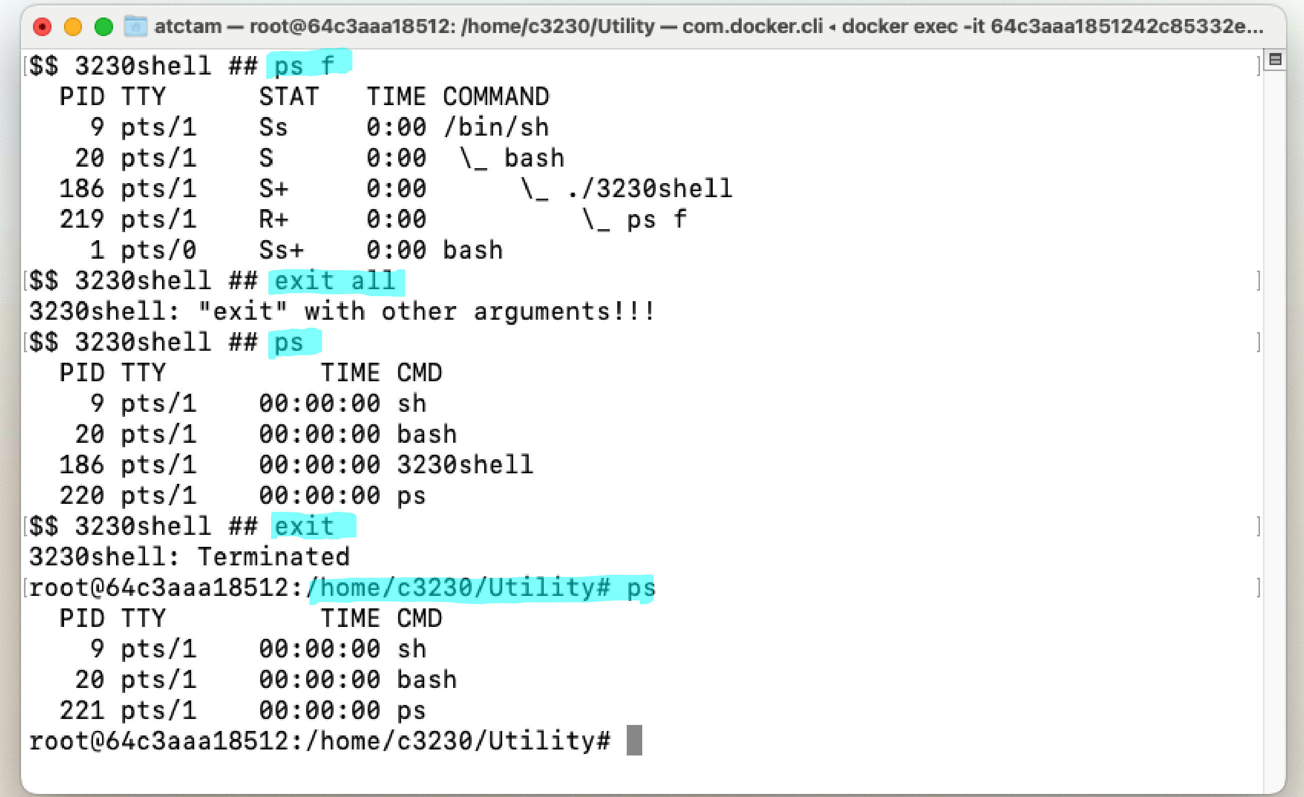


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## cat 3230shell.c | wc -l
576
$$ 3230shell ## ps f
  PID TTY      STAT   TIME COMMAND
  248 pts/1    Ss     0:00 /bin/sh
  254 pts/1    S      0:00  \_ bash
  336 pts/1    S+     0:00      \_ ./3230shell
  348 pts/1    R+     0:00          \_ ps f
    1 pts/0    Ss+    0:00 bash
$$ 3230shell ## cat PASS1.txt | grep shell | wc -l
95
$$ 3230shell ## cat Makefile | wc -l
34
$$ 3230shell ## cat Makefile | ./relay | ./relay | wc -l
34
$$ 3230shell ## cat PASS1.txt | ./relay | ./relay 1 | wc -l
234
$$ 3230shell ## cat Makefile | | wc -l
3230shell: should not have two consecutive | without in-between command
$$ 3230shell ##
```

# Feature - exit

- Built-in command: *exit* - to terminate the 3230shell program

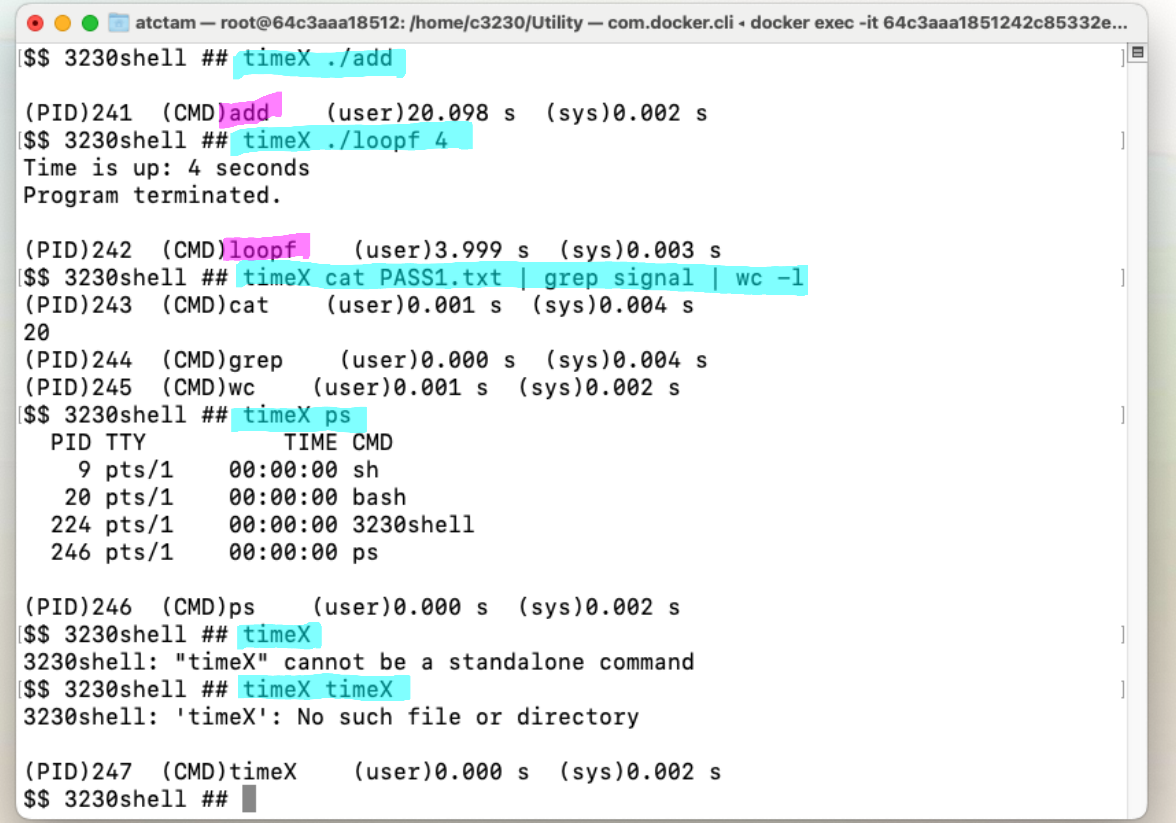


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## ps f
  PID TTY      STAT   TIME COMMAND
    9 pts/1    Ss     0:00 /bin/sh
   20 pts/1    S      0:00  \_ bash
  186 pts/1    S+     0:00      \_ ./3230shell
  219 pts/1    R+     0:00          \_ ps f
    1 pts/0    Ss+    0:00 bash
$$ 3230shell ## exit all
3230shell: "exit" with other arguments!!!
$$ 3230shell ## ps
  PID TTY          TIME CMD
    9 pts/1    00:00:00 sh
   20 pts/1    00:00:00 bash
  186 pts/1    00:00:00 3230shell
  220 pts/1    00:00:00 ps
$$ 3230shell ## exit
3230shell: Terminated
root@64c3aaa18512:/home/c3230/Utility# ps
  PID TTY          TIME CMD
    9 pts/1    00:00:00 sh
   20 pts/1    00:00:00 bash
  221 pts/1    00:00:00 ps
root@64c3aaa18512:/home/c3230/Utility# 
```

# Feature - timeX

- Built-in command: *timeX* - the user uses this command to find out the process statistics of all terminated child process(es) under that input command line

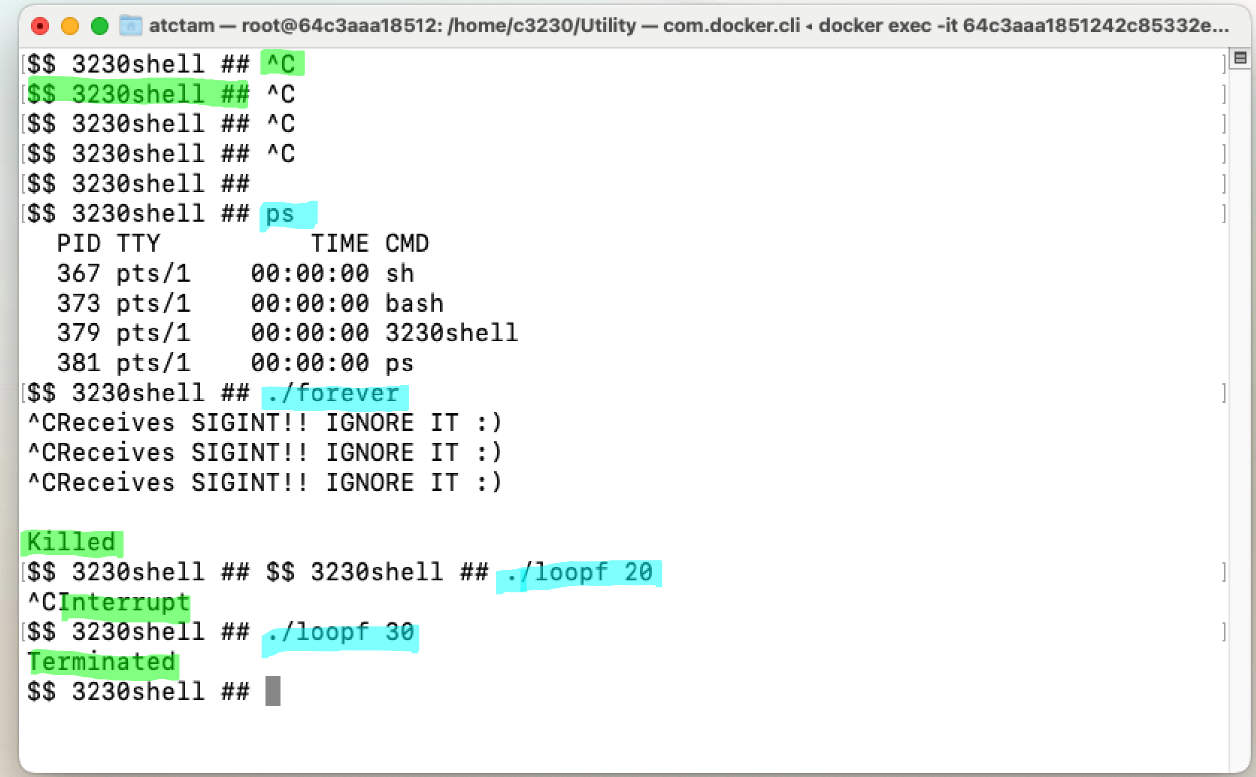


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## timeX ./add

(PID)241  (CMD)add    (user)20.098 s  (sys)0.002 s
$$ 3230shell ## timeX ./loopf 4
Time is up: 4 seconds
Program terminated.

(PID)242  (CMD)loopf    (user)3.999 s  (sys)0.003 s
$$ 3230shell ## timeX cat PASS1.txt | grep signal | wc -l
(PID)243  (CMD)cat    (user)0.001 s  (sys)0.004 s
20
(PID)244  (CMD)grep    (user)0.000 s  (sys)0.004 s
(PID)245  (CMD)wc    (user)0.001 s  (sys)0.002 s
$$ 3230shell ## timeX ps
  PID TTY          TIME CMD
    9 pts/1    00:00:00 sh
   20 pts/1    00:00:00 bash
  224 pts/1    00:00:00 3230shell
  246 pts/1    00:00:00 ps

(PID)246  (CMD)ps    (user)0.000 s  (sys)0.002 s
$$ 3230shell ## timeX
3230shell: "timeX" cannot be a standalone command
$$ 3230shell ## timeX timeX
3230shell: 'timeX': No such file or directory

(PID)247  (CMD)timeX    (user)0.000 s  (sys)0.002 s
$$ 3230shell ##
```

# Feature – SIGINT signal handling

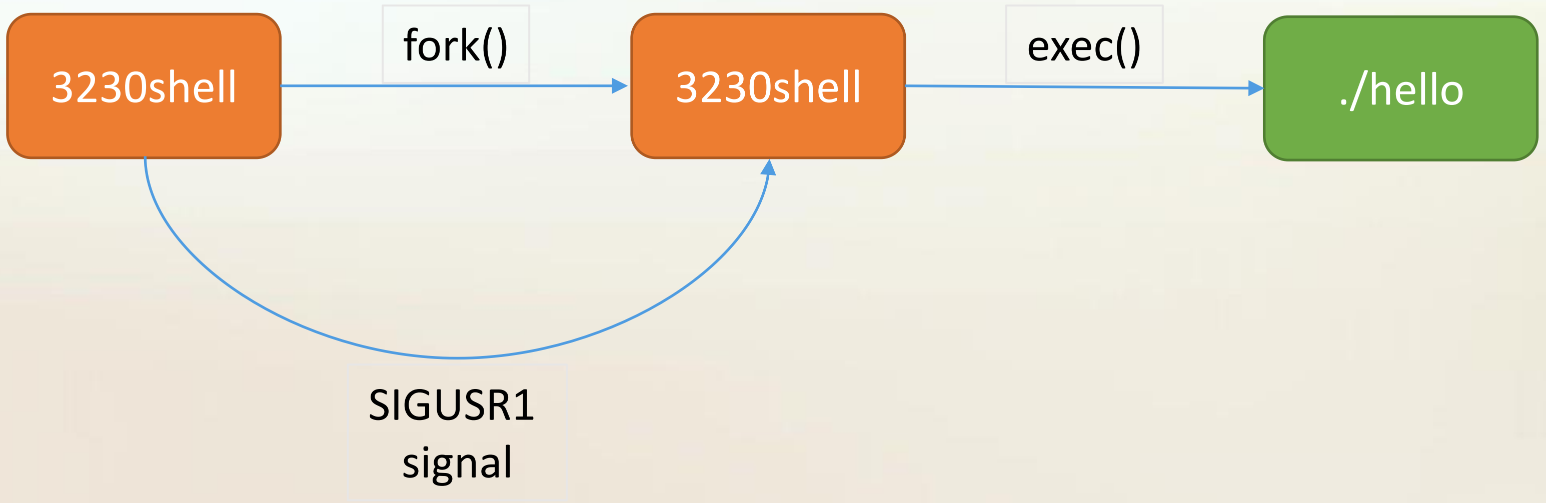
- 3230shell should not be terminated by SIGINT (Cltr-c) signal.
- Foreground job should response to SIGINT signal if the signal is detected; whether the foreground job will terminate or not should depend on the defined logic in that program
- If a process is terminated by signal, displays the signal information


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
[$$ 3230shell ## ^C
[$$ 3230shell ## ^C
[$$ 3230shell ## ^C
[$$ 3230shell ## ^C
[$$ 3230shell ##
[$$ 3230shell ## ps
  PID TTY          TIME CMD
  367 pts/1    00:00:00 sh
  373 pts/1    00:00:00 bash
  379 pts/1    00:00:00 3230shell
  381 pts/1    00:00:00 ps
[$$ 3230shell ## ./forever
^CReceives SIGINT!! IGNORE IT :)
^CReceives SIGINT!! IGNORE IT :)
^CReceives SIGINT!! IGNORE IT :)

Killed
[$$ 3230shell ## $$ 3230shell ## ./loopf 20
^CInterrupt
[$$ 3230shell ## ./loopf 30
Terminated
$$ 3230shell ## 
```

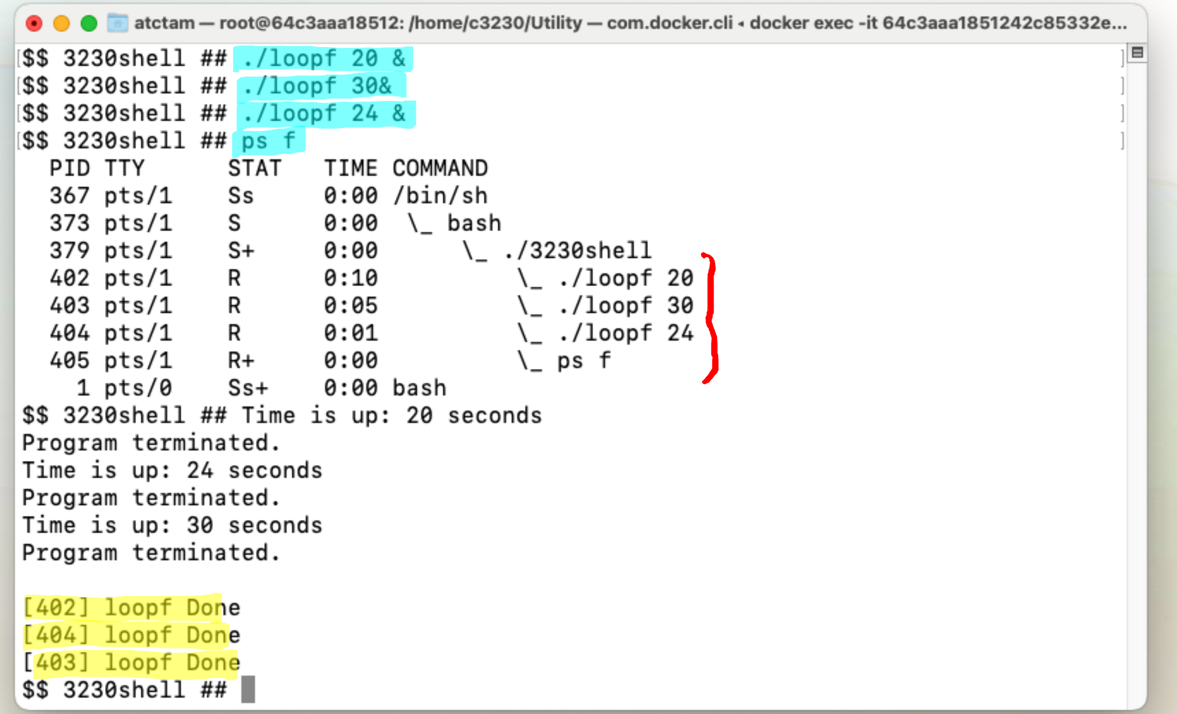
## Feature – SIGUSR1 signal

- Uses SIGUSR1 signal for controlling when the child process starts running the target command



# Bonus Feature - & operator

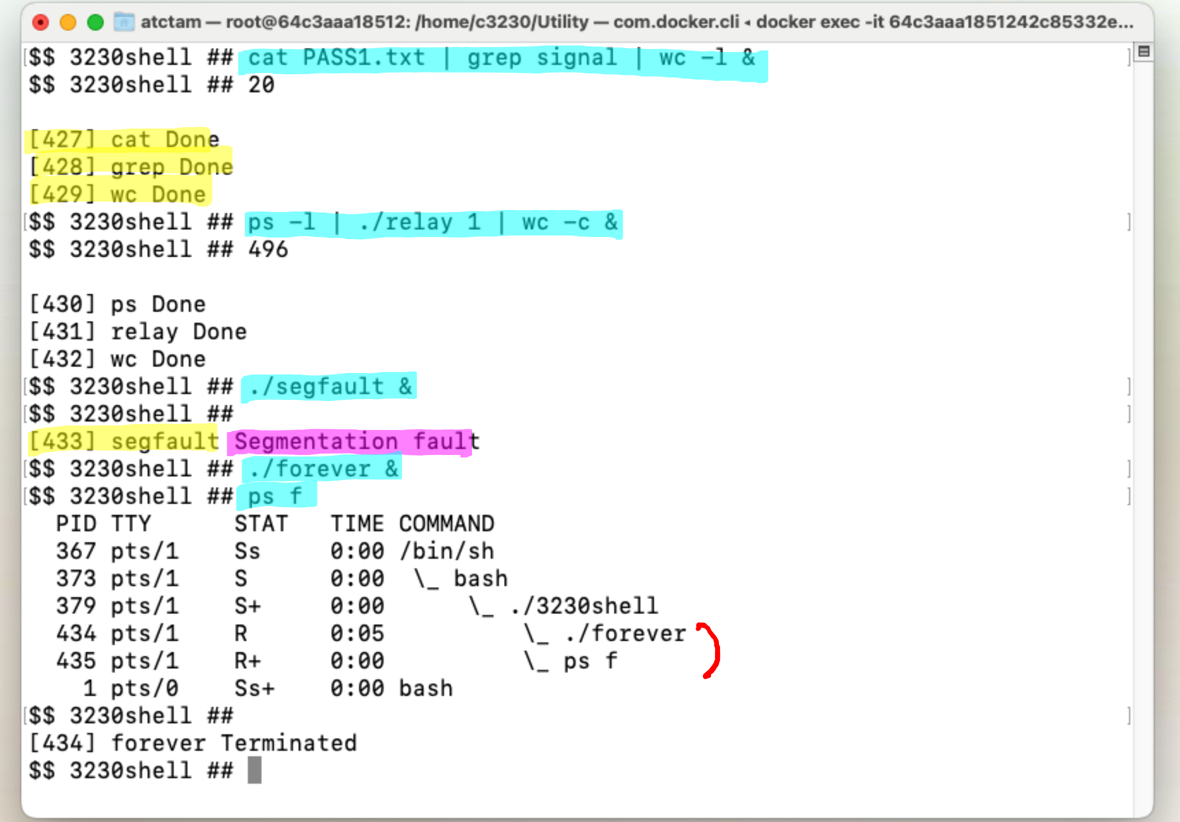
- When the last character is the & character, the whole command(s) will be executed in background; the shell prompt will be displayed immediately.
- Termination of a background child process is detected that by means of the SIGCHLD signal
- 3230shell prints out a statement to indicate that it has detected the termination of a background child process


atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## ./loopf 20 &
$$ 3230shell ## ./loopf 30&
$$ 3230shell ## ./loopf 24 &
$$ 3230shell ## ps f
  PID TTY      STAT   TIME COMMAND
  367 pts/1    Ss     0:00 /bin/sh
  373 pts/1    S      0:00  \_ bash
  379 pts/1    S+     0:00      \_ ./3230shell
  402 pts/1    R      0:10          \_ ./loopf 20
  403 pts/1    R      0:05          \_ ./loopf 30
  404 pts/1    R      0:01          \_ ./loopf 24
  405 pts/1    R+     0:00          \_ ps f
    1 pts/0    Ss+    0:00 bash
$$ 3230shell ## Time is up: 20 seconds
Program terminated.
Time is up: 24 seconds
Program terminated.
Time is up: 30 seconds
Program terminated.

[402] loopf Done
[404] loopf Done
[403] loopf Done
$$ 3230shell ##
```

# Bonus Feature – SIGCHLD signal

- 3230shell should remove all zombie processes as soon as the processes have terminated
- For each terminated background process, 3230shell should output a statement indicating the termination status of the process
- The statement(s) will be displayed only when the next shell prompt reappears



atctam — root@64c3aaa18512: /home/c3230/Utility — com.docker.cli ◂ docker exec -it 64c3aaa1851242c85332e...

```
$$ 3230shell ## cat PASS1.txt | grep signal | wc -l &
$$ 3230shell ## 20

[427] cat Done
[428] grep Done
[429] wc Done
$$ 3230shell ## ps -l | ./relay 1 | wc -c &
$$ 3230shell ## 496

[430] ps Done
[431] relay Done
[432] wc Done
$$ 3230shell ## ./segfault &
$$ 3230shell ##
[433] segfault Segmentation fault
$$ 3230shell ## ./forever &
$$ 3230shell ## ps f
  PID TTY      STAT   TIME COMMAND
  367 pts/1    Ss     0:00 /bin/sh
  373 pts/1    S      0:00  \_ bash
  379 pts/1    S+     0:00      \_ ./3230shell
  434 pts/1    R      0:05          \_ ./forever
  435 pts/1    R+     0:00          \_ ps f
    1 pts/0    Ss+    0:00 bash
$$ 3230shell ##
[434] forever Terminated
$$ 3230shell ##
```

## Computer platform to use

- You are expected to develop your program
  - Using Windows WSL2 with Ubuntu app;    Or
  - Using Ubuntu docker image; Or
  - Using a native Ubuntu installation on your notebook/laptop; Or
  - Using the Dept's Linux server *workbench2.cs.hku.hk or academy11.cs.hku.hk or academy21.cs.hku.hk*
- After fully tested locally, upload the program to the workbench2 for the final tests
- Your programs must be written in C and successfully compiled with gcc
  - Make sure that your program can be compiled ***without any errors***

# Subdivide the task into Four stages

- Stage 1 – support one command (with arguments) only
  - parse the command line
  - create new process to execute a valid command
  - wait for child process to terminate and get its exit status
  - print terminated child process's statistics
- Stage 2
  - handle SIGINT signal correctly
  - use SIGUSR1 signal for control
  - implement the exit command
- Stage 3 - support pipes
  - parse the command line to get more than one commands separate by '|' sign
  - use pipe() and dup2() system calls to set up a pipe between two processes
  - wait for child processes to terminate and get their exit status
- Stage 4
  - Complete the final 3230shell program with the remaining features

Lab 1

Lab 2

## Some Utility Programs

- Utility.zip
  - Contains some useful C programs for testing certain features of the shell program

```
add-loop.c
cmdarg.c
inProc.c
loop.c
loopever.c
Makefile
outProc.c
PASS1.txt
relay.c
segfault.c
start.txt
```

## Submission of Assignment

- You should name your program 3230shell_StudentNumber.c
- Submit your programs to the course's Moodle web site
- Add your signature in the header of your submitted program and include clear comments

```
/*************************************************************
* Student name and No.: Harry Potter 3015999999
* Development platform: Ubuntu docker container
* Remark: Complete the basic without bonus
  features
*************************************************************/
```

# Grading Rubric

- For the details, please read project specification document

| | |
|---|---|
| **Documentation (1 point)** | Student's info |
| | Sufficient comments |
| **Correctness of the program (11 points)** | Process creation and execution – foreground (3 points) |
| | Process creation and execution – use of '\|' (3 points) |
| | Use of signals (2.5 points) |
| | Built-in command: timeX (2 points) |
| | Built-in command: exit (0.5 points) |
| **Bonus part (4 points)** | Process creation and execution - background (2 points) |
| | SIGCHLD signal (2 points) |