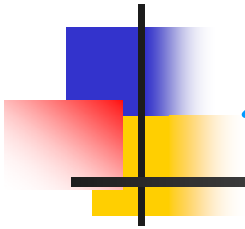
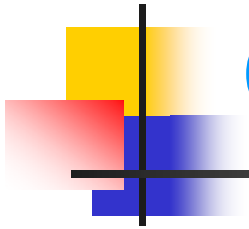
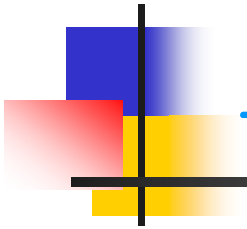


제9장 프로세스 관계

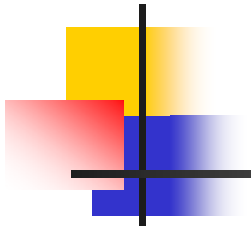


Contents

1. Logins
2. Process Groups
3. Sessions
4. Controlling Terminal
5. Job Control



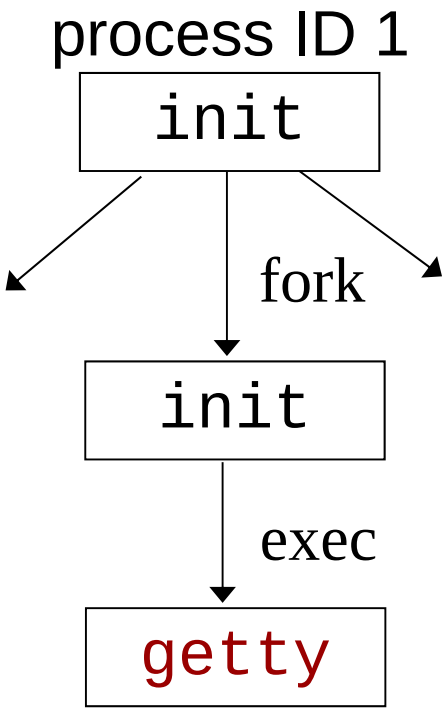
로그인



터미널 로그인

- `/etc/ttys`: 1 line per terminal device
- `getty`: opens terminal device for reading and writing
- `gettytab`: data file for `getty`
- `login`: login program
- `getpass`: read password
- `crypt`: encrypts plain text (password)

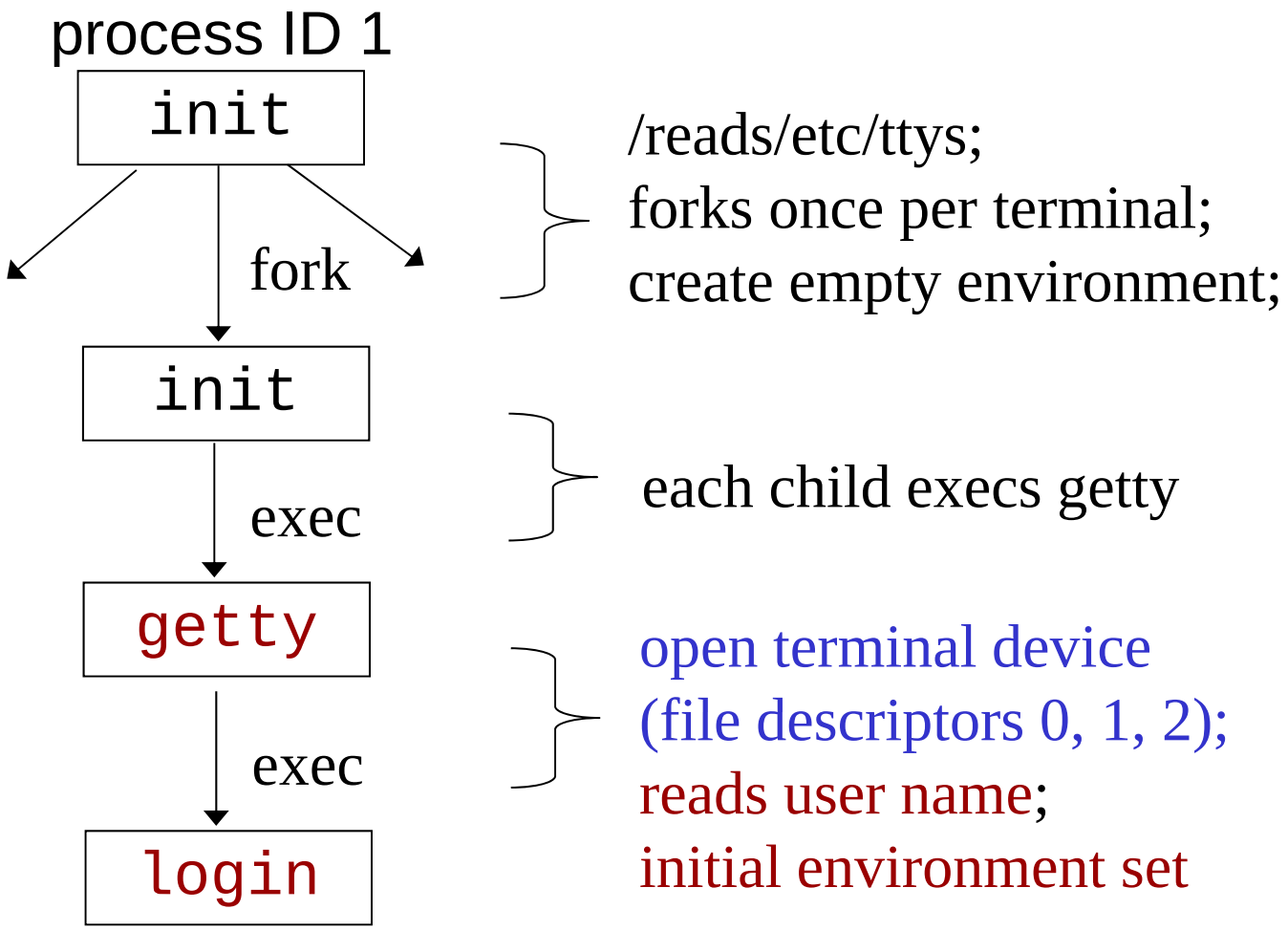
터미널 로그인 과정

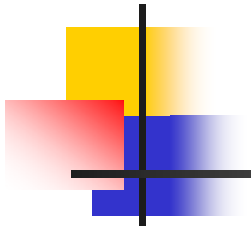


/reads/etc/ttys;
forks once per terminal;
create empty environment;

each child execs getty

로그인 후 상태

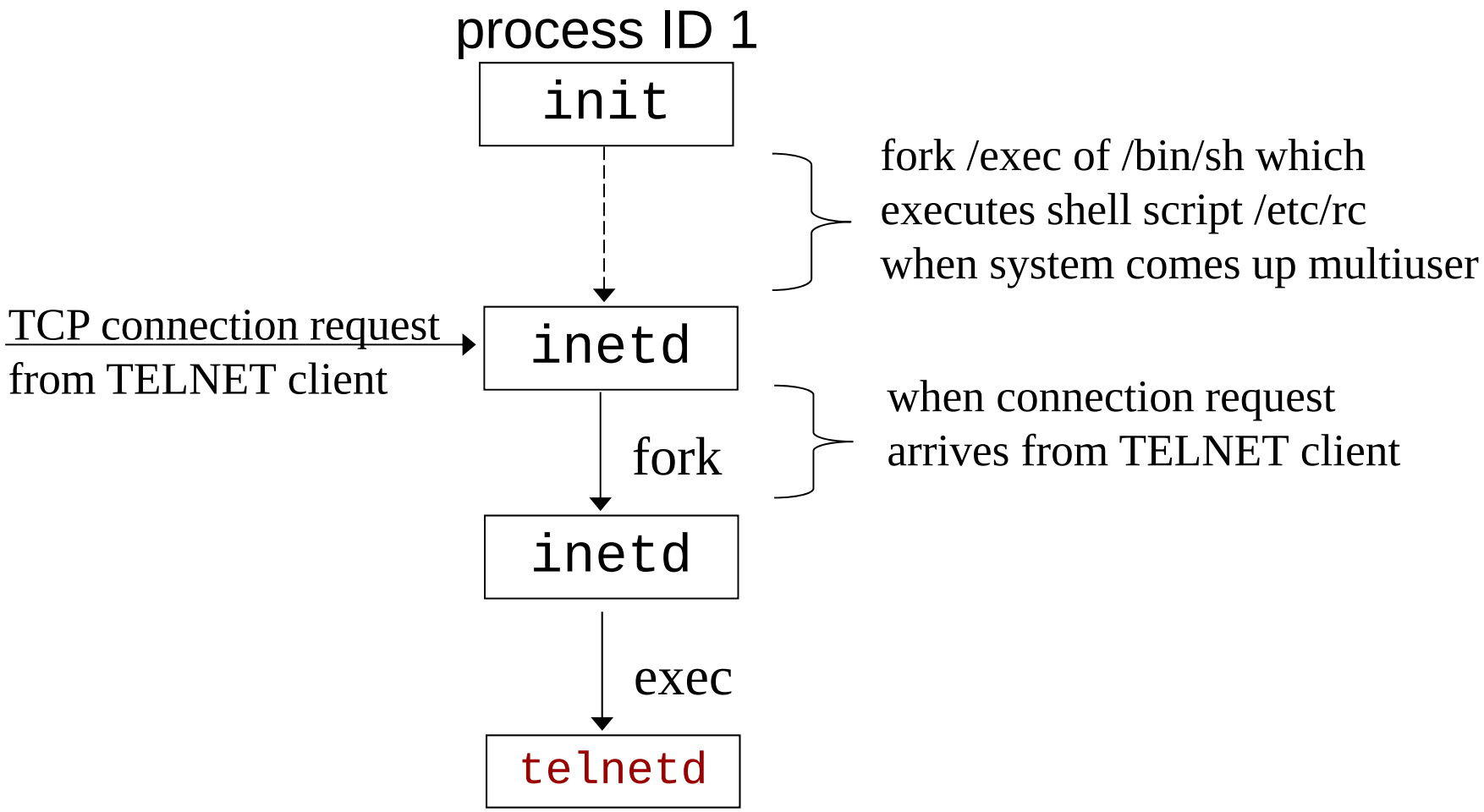

process ID 1
init
fork
init
exec
getty
exec
login
/reads/etc/ttys;
forks once per terminal;
create empty environment;
each child execs getty
open terminal device
(file descriptors 0, 1, 2);
reads user name;
initial environment set

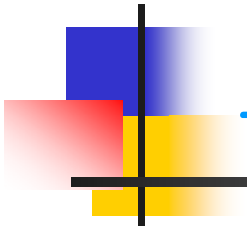


네트워크 로그인

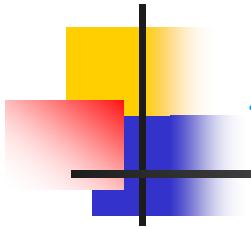
- inetd – Internet superserver
  - waits for TCP/IP connection requests
  - when request arrives, does a fork and exec of the appropriate program
- telnet
  - a remote login application(client) that uses TCP/IP protocol.
- telnetd
  - TELNET server
  - opens pseudo-terminal device
  - splits into two processes using fork
    - The parent handles the communication across the network connection
    - The child does an exec of the login program.

TELNET 서버


process ID 1
init
fork /exec of /bin/sh which
executes shell script /etc/rc
when system comes up multiuser
TCP connection request
from TELNET client
inetd
when connection request
arrives from TELNET client
fork
inetd
exec
telnetd



프로세스 그룹

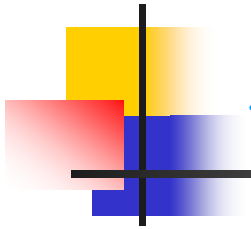


프로세스 그룹

- 프로세스 IDs
  - Process ID(PID)
  - Process Group ID(GID)
  - 각 프로세스는 하나의 프로세스 그룹에 속함.
  - 각 프로세스는 자신이 속한 process group ID를 가지며
  - fork 시 물려받는다.

```
#include <sys/types.h>
#include <unistd.h>

pid_t getpgrp(void);
Returns: process GID of calling process
```

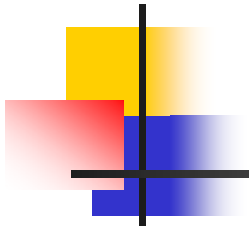


프로세스 그룹

- 프로세스 그룹이란?
  - 통상 한 부모(그룹 리더)가 생성하는 자손 프로세스들이 하나의 프로세스 그룹을 형성한다.
  - Each process group has a process group leader
    Process GID = PID
- 프로세스 그룹은 무엇을 하는데 사용되나요?
  - 프로세스 그룹은 주로 signal 전달 등을 위해 설정됨
  - kill -INT -3245 명령은 프로세스 그룹 3245에 SIGINT보냄
  - kill -INT 3245 명령은 프로세스 3245에만 SIGINT 보냄

프로세스 그룹: 예

```
#include <sys/types.h>
#include <unistd.h>
main()
{
    int pid, gid;
    printf("PARENT: PID = %d GID = %d\n", getpid(), getpgrp());
    pid = fork();
    if (pid == 0) {  // 자식 프로세스
        printf("CHILD: PID = %d GID = %d\n", getpid(), getpgrp());
    }
}
```



프로세스 그룹

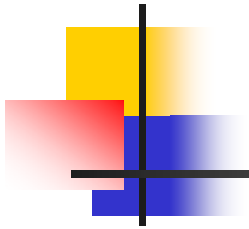
- 프로세스 그룹 만들기
  - 자식 프로세스는 setpgrp 호출을 통하여 기존의 프로세스 그룹에서 벗어나서 새로운 프로세스 그룹을 형성할 수 있다.
  - A process can create a new process group and become leader
  - int setpgid(pid_t *pid*, pid_t *pgid*);
  - A process group leader can create processes in the group
- 프로세스 그룹 소멸
  - the last process terminates OR
  - joins another process group
  - (leader may terminate first)

Process Groups

```
#include <sys/types.h>
#include <unistd.h>

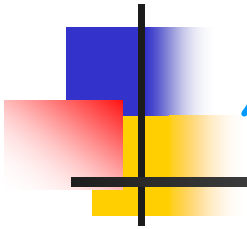
int setpgid(pid_t pid, pid_t pgid);
                                    Returns: 0 if OK, -1 on error
```

- 역할
  - Create new group or join existing group.
- Set the process GID of the process *pid* to *pgid*.
  - *pid* == *pgid* → leader
  - *pid* != *pgid* → pid becomes a member of the process group
  - *pid* == 0 → PID of caller 사용
  - *pgid* == 0 → *pid* 가 process group leader 됨
- Process can set PGID of itself or its children



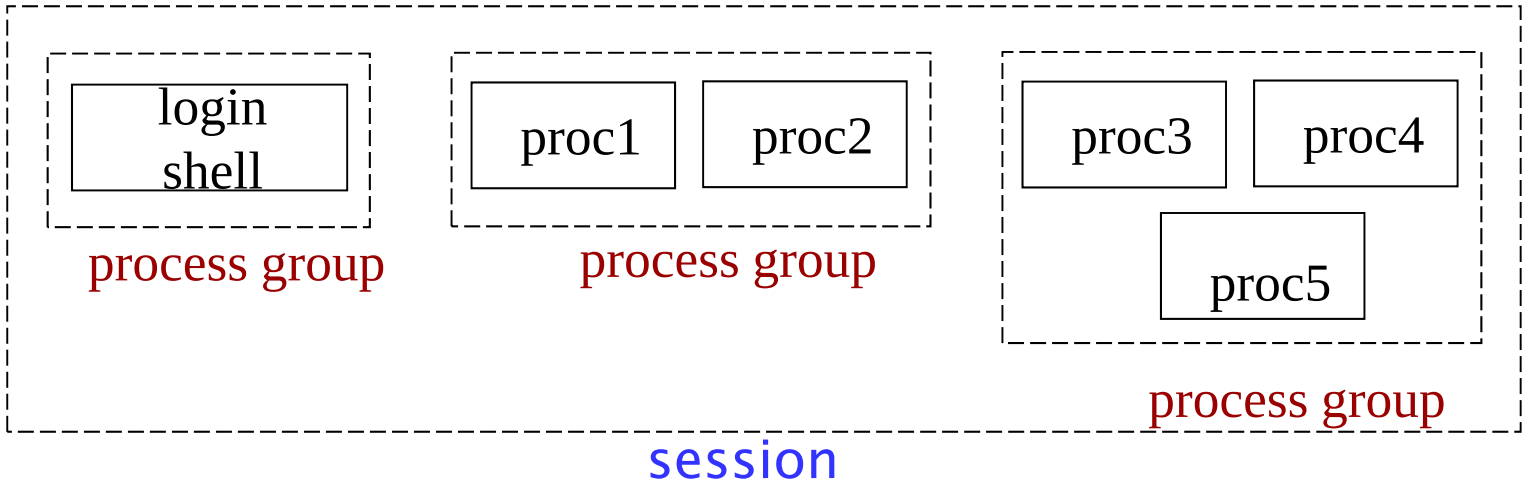
Example

```
#include <sys/types.h>
#include <unistd.h>
main()
{
    int pid, gid;
    printf("PARENT: PID = %d  GID = %d \n", getpid(), getpgrp());
    pid = fork();
    if (pid == 0) {
        setpgid(0, 0);
        printf("CHILD: PID = %d  GID = %d \n", getpid(), getpgrp());
    }
}
```



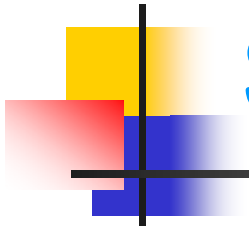
세션/컨트롤 터미널

세션



- 세션이란?
  - Collection of one or more process groups
  - Job 제어에 사용된다.
- 예

```
$proc1 | proc2 &
$proc3 | proc4 | proc5
```

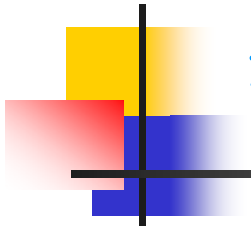


Sessions

```
#include <sys/types.h>
#include <unistd.h>

pid_t setsid(void);
                                Returns: PGID if OK, -1 on error
```

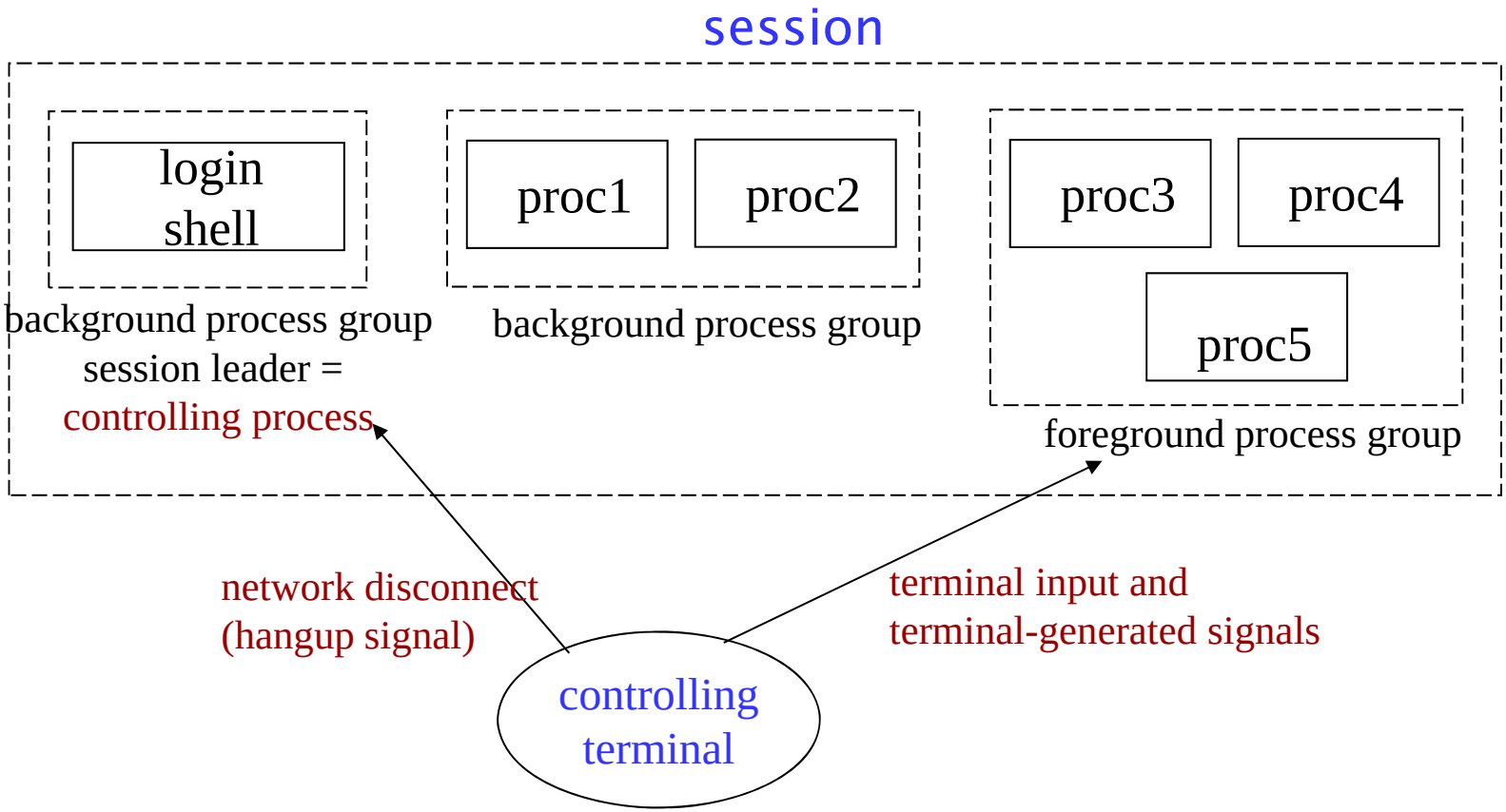
- 새로운 세션 만들기
  - A process (not a process group leader) creates a new session
  - The process becomes session leader (new session contains this process only)
  - The process become process group leader of a new process group (PGID=PID)



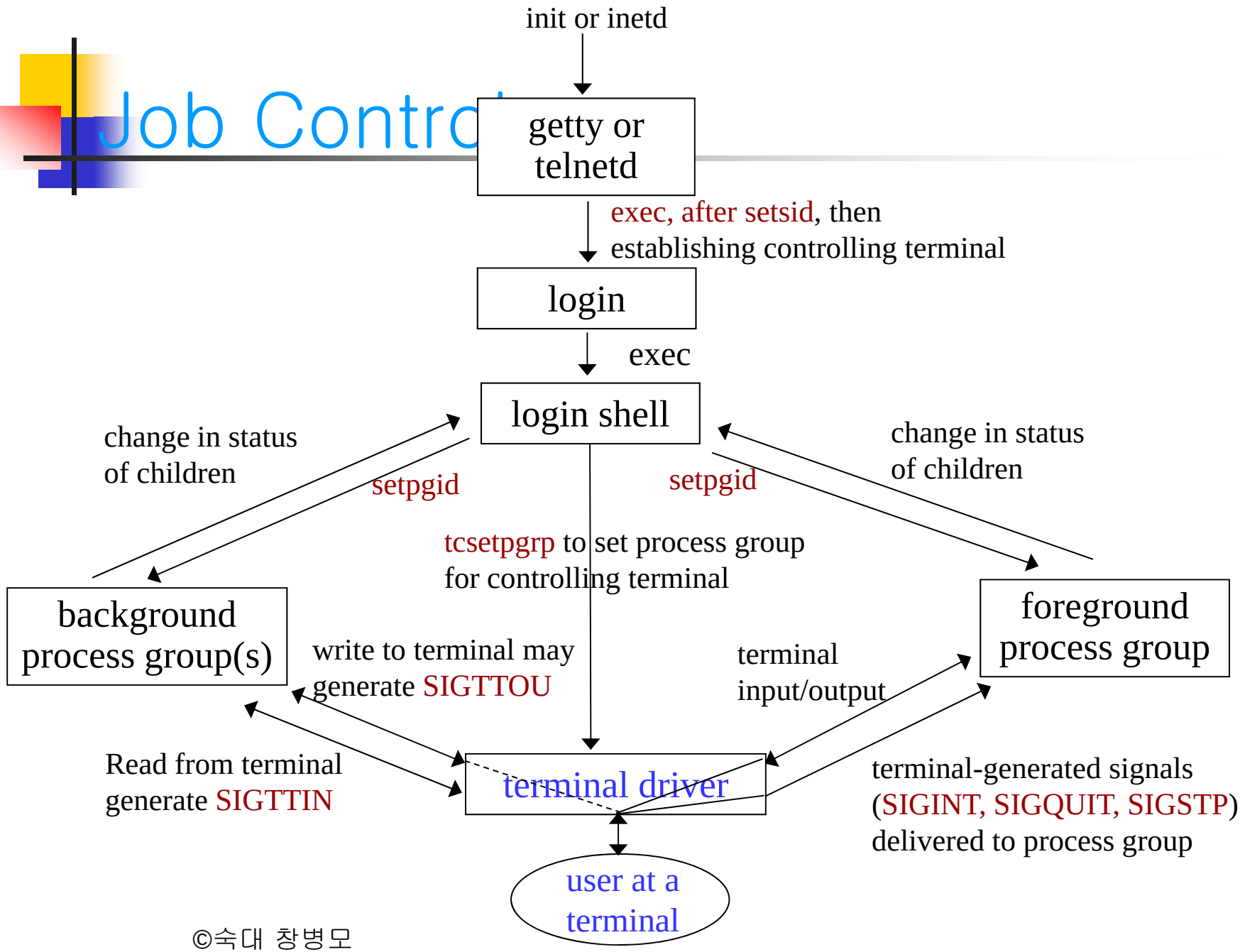
컨트롤 터미널

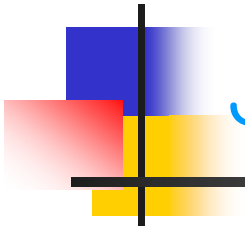
- 세션을 컨트롤할 수 있는 터미널
  - A session can have a controlling terminal
  - The session leader is the controlling process
- 세션 내의 프로세스 그룹들
  - 1 foreground,
  - 1 or more backgrounds
- 이 foreground group이 입력을 받는다.
  - 컨트롤 터미널로부터 키보드 입력 및 signal 받음
  - Interrupt key from control terminal
    → SIGINT to all foreground processes

컨트롤 터미널

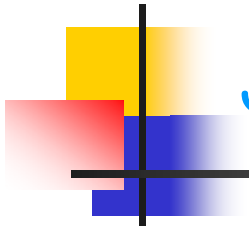

session
login shell
background process group
session leader =
controlling process
proc1
proc2
background process group
proc3
proc4
proc5
foreground process group
network disconnect
(hangup signal)
terminal input and
terminal-generated signals
controlling terminal

Job Contro


init or inetd
getty or telnetd
exec, after setsid, then establishing controlling terminal
login
exec
login shell
change in status of children
setpgid
setpgid
change in status of children
tcsetpgrp to set process group for controlling terminal
background process group(s)
foreground process group
write to terminal may generate SIGTTOU
terminal input/output
Read from terminal generate SIGTTIN
terminal driver
terminal-generated signals (SIGINT, SIGQUIT, SIGSTP) delivered to process group
user at a terminal

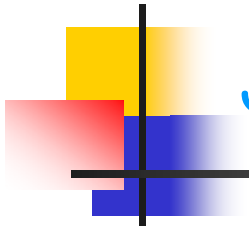


Job Control



Job Control

- A feature added by Berkeley in 1980
- Job
  - A group of processes
- To start multiple jobs from a single terminal
- To control:
  - which jobs can access terminal
  - which jobs are to run in background.



Job Control

- 3 forms of support required by job control:
  - A shell to support job control
  - Terminal driver must support job control
  - Job-control signals

- Example
  - `cat /etc/passwd | grep faculty | sort > faculty.out &`
  - `ls -l | grep test | sort > test.out &`

  ← 2 background jobs

  - `vi main.c` ← 1 foreground job

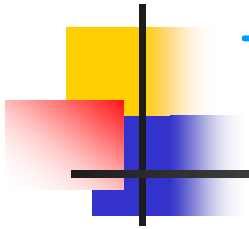
Shell for Job Control

- Shells assign a job ID to a background job

- ```
  $cat /etc/passwd | grep faculty | sort > faculty.out &
  [1] 19277 19278 19279
  $ls -l | grep test | sort > test.out &
  [2] 19280 19281 19282
  $
  [2] +  Done      cat /etc/passwd | grep  …  &
  [1] +  Done      ls -l | grep test | … &
  ```

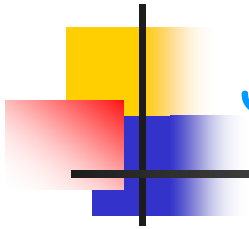
process IDs

job number
  ```



Terminal Driver for Job Control

- Terminal driver looks out for 3 special characters:
  - Interrupt character (DELETE or CTRL-C) generates SIGINT
  - Quit character (CTRL-BACKSLASH) generates SIGQUIT
  - Suspend character (CTRL-Z) generates SIGTSTP
- SIGTTIN
  (1) When background job tries to read from the terminal,
  (2) the terminal driver sends it to the background job
  (3) This stops the background job.
- SIGTTOU (stty tostop 했을 경우에)
  (1) When background job tries to write to the terminal,
  (2) the terminal driver sends it to the background job
  (3) This stops the background job.

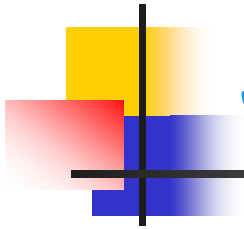


Job Control

```
$ cat > temp.foo &
  [1] 1719
$
  [1] + Stopped(tty 입력) cat>temp.foo &

$ fg %1
cat> temp.foo
hello, world
 ^D

$cat temp.foo
  hello, world
```



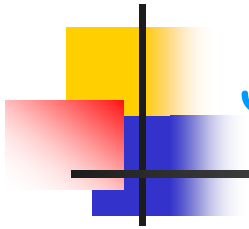
Job Control

```
$ cat temp.foo &
[1] 1719
$ hello, world

[1] + Done cat temp.foo &

$ stty tostop
$ cat temp.foo &
[1] 1721
$
[1] + Stopped(tty output) cat temp.foo &

$ fg %1
cat temp.foo
hello, world
```

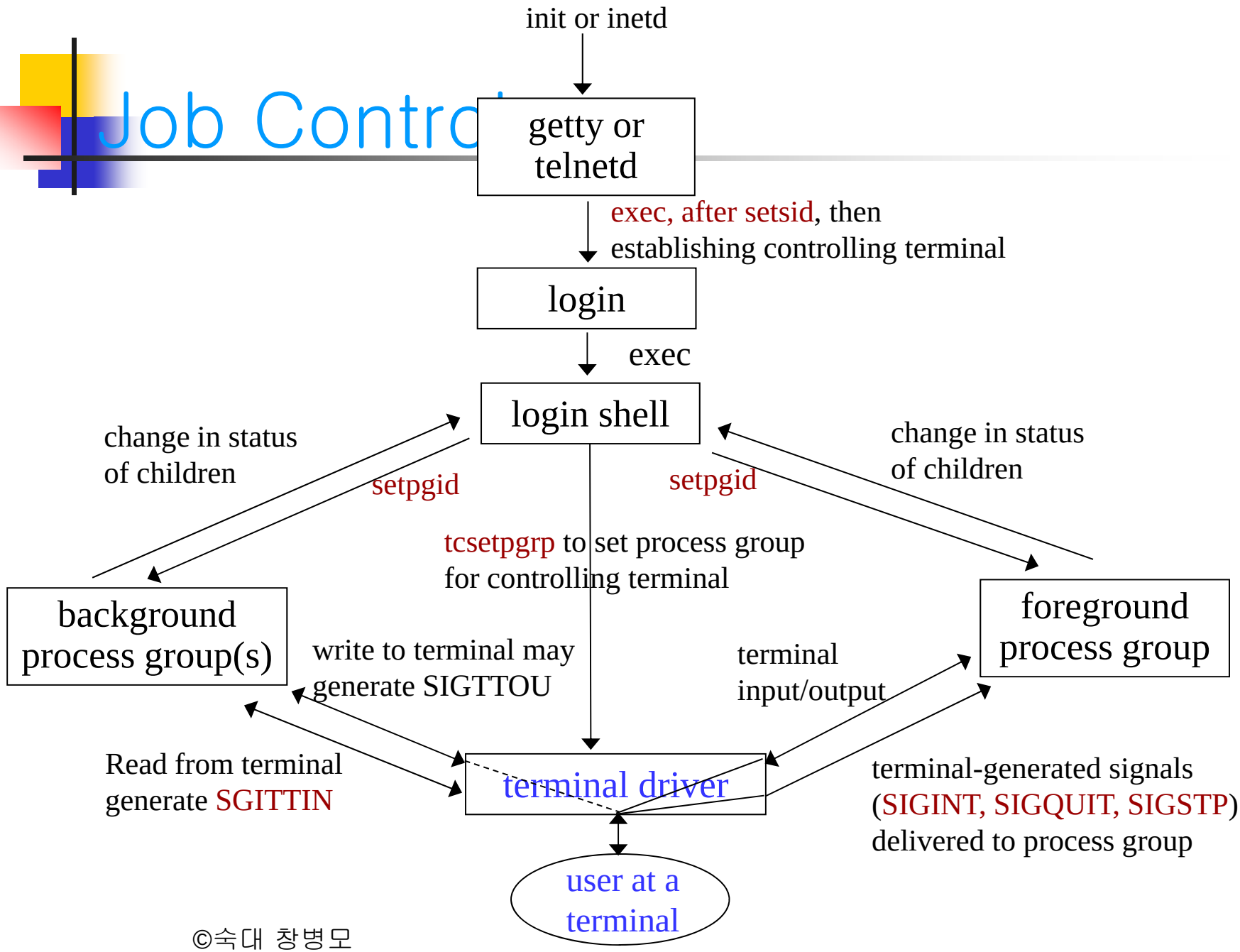


Job Control

- fg command
  - The shell places the job into the foreground process group, and
  - sends **SIGCONT** to the process group
- stty command
  - stty tostop

    disable ability of background jobs to output to the controlling terminal

Job Control


init or inetd
getty or telnetd
exec, after setsid, then establishing controlling terminal
login
exec
login shell
change in status of children
setpgid
setpgid
change in status of children
tcsetpgrp to set process group for controlling terminal
background process group(s)
foreground process group
write to terminal may generate SIGTTOU
terminal input/output
Read from terminal generate SGITTIN
terminal driver
terminal-generated signals (SIGINT, SIGQUIT, SIGSTP) delivered to process group
user at a terminal