Assignment7 Report

1

1.1 Implementation Details

Why the Child Process Does Not Have a Controlling Terminal:

When a new session is created by the child process using setsid(), the child becomes the session leader of this new session. As part of this, the child process also loses its controlling terminal if it had one. This is because sessions are designed to be independent entities in terms of controlling terminals, allowing for background processes and daemons that do not need user interaction. PID, PGRP, and TPGID Values:

PID (Process ID): Unique identifier for each process. PGRP (Process Group ID): ID of the process group. A process group is a collection of one or more processes, usually associated with the same job, that can receive signals collectively. TPGID (Terminal Process Group ID): ID of the foreground process group of the controlling terminal. If a process does not have a controlling terminal, this will be -1. In the case of our child process:

The PID will be the unique identifier for the child process. The PGRP will be the same as the child's PID since it becomes a process group leader. The TPGID will be -1, indicating no controlling terminal. Compile and run this program in a Unix-like environment to observe the behavior. Make sure to have necessary permissions and configurations to execute system calls like fork() and setsid().

1.2 Codes

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    pid_t pid = fork();
    if (pid == 0) {
        // child process
        pid_t sid = setsid();
        if (sid < 0) {
            printf("Error: cannot create a new session\n");
            return -1;
        }

        printf("child process pid: %d, pgid: %d, sid: %d\n", getpid(),
            getpgid(getpid()), sid);

        FILE *fp;
        char str[1024];
        char real_cmd[1024];
        char *cmd = "ps -x -o pid,pgid,tpgid";
        sprintf(real_cmd, cmd, getpgid(getpid()));

        fp = popen(real_cmd, "r");
        if (fp == NULL) {
            printf("popen failed\n");
            return -1;
        }
        while (fgets(str, sizeof(str), fp) != NULL) {
            printf("%s", str);
        }
        pclose(fp);
    } else if (pid > 0) {
        // parent process
        // semaphore wait
        wait(NULL);
    } else {
        perror("Error: cannot fork a new process\n");
        return -1;
    }
    return 0;
}
```

程式碼 **1.1:** assignment7.c

```
# One FreeBSD 13.2-RELEASE-p4 FreeBSD 13.2-RELEASE-p4 GENERIC amd64
ryanchang1117@freebsd-13-1:~/APUE_assignment7 $ ./assignment7

child process pid: 1044, pgid: 1044, sid: 1044
 PID PGID TPGID
 902  900     0
1044 1044     0
1045 1044     0
 903  903  1043
1043 1043  1043
```

程式碼 **1.2:** 指令紀錄 (On BSD)

```
# On Linux 5.4.0-166-generic #183-Ubuntu SMP Mon Oct 2 11:28:33 UTC 2023
    x86_64 GNU/Linux
hsuan@t1:~/APUE_assignment7$ ./assignment7
child process pid: 767208, pgid: 767208, sid: 767208
    PID    PGID   TPGID
 766697  766697      -1
 766699  766697      -1
 766706  766706      -1
 766725  766725      -1
 766804  766675      -1
 766805  766805  767207
 767199  767199  767207
 767207  767207  767207
 767208  767208      -1
 767209  767208      -1
 767210  767208      -1
```

程式碼 **1.3:** 指令紀錄 (On Ubuntu)

```
CC = gcc
CFLAG = -std=c11 -O2 -Wall
TARGET = assignment7
SRCS = assignment7.c
OBJS = assignment7.o

RPT_FILES := report.tex
PDF_FILES := report.pdf

all: clean $(TARGET)

$(TARGET): $(OBJS)
  $(CC) $(CFLAG) -o $(TARGET) $(OBJS)

%.o: %.c
  $(CC) $(CFLAGS) -c $< -o $@

- pdf:
  docker run -v $(shell pwd):/code -it --rm --name xelatex-build lfswang/
      xelatex:latest sh /code/run.sh

clean:
  rm -f $(OBJS) $(TARGET) *.toc *.synctex.gz *.out *.log *.aux *.lot *.
      lof *.bcf *.run.xml *.pdf
```

程式碼 **1.4:** Makefile