- # Zadanie 1

(5pt) Wytłumacz jakie pliki zawierają katalogi **/dev** oraz **/proc**. Wykorzystując polecenie **dd** odczytaj pierwszy sektor z dysku głównego (uwaga na prawa dostępu) lub podpiętego pendrive'a i wyświetl przez **hexdump -C**. Z katalogu **proc** wyświetl informacje o pamięci, procesorze i partycjach.

1. Katalogi:

- **/dev -** od devices, znajdujące się w nim pliki to pliki pośredniczące w komunikacji system- urządzenia. Nie są to faktyczne pliki na dysku.
- **/proc –** od processes, jest katalogiem wirtualnym, znajdują się w nim dane o aktualnie uruchomionych procesach

2. Sektor i pendrive hexump
    - *pierwszy sektor dysku głównego*

    sudo dd if=/dev/nvme0n1 bs=1 count=1k | hexdump -C

    - *pendrive*

    sudo dd if=/dev/disk/by-id/usb-Kingston_DataTraveler_3.0_1831BFBBBED1B310A95F0050-0\:0-part1 bs=1 count=1k | hexdump -C

3. Informacje z katalogu *proc* ***cd proc*** **>** ***ls - l | grep partition/ mem/ cpu*** **> cat**
    - *partycje* **– cat proc/partitions**
    - *pamięć* **- cat proc/meminfo**
    - *procesor* – **cat proc/cpuinfo**

- # Zadanie 2

(5pt) Zapoznaj się z programem **ps** (**man ps**). Naucz się posługiwać tym programem, aby umieć sprawdzać co najmniej istnienie i parametry wybranych procesów (PID procesu i rodzica, stan procesu, priorytet, nice, ilość pamięci, zużycie czasu procesora). Uruchom też kilka terminali pokaż jakie urządzenie tty wykorzystują. Wykonując komendę **ps axjf** pokaż wszystkie procesy które podpięte są do tych terminali (kolumna TTY).

1. **Ps** – wyświetla informacje o wyselekcjonowanych aktywnych procesach.
2. Pid po nazwie **Ps -C nazwa**
3. ppid po pid X **ps -o ppid=,tty= -p X**
4. pid po ppid **ps -f --ppid X**
5. sort **ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head**
6. morderstwo procesu **kill -9 2583**
7. priorytet **ps -p 8238 -o priority=**
8. nice **ps -p 8238 -o nice=**
9. podpięcia **ps-t**
10. pamięć i zużycie cpu **ps -p 4445 -o %mem=,%cpu=**
11. select tylko konkretnego terminalu **ps axjf |grep pts/3**

- # Zadanie 3

Wytłumacz każdy z powyższych kroków. Co oznaczają opcje **-Wall** oraz **-pedantic**? Zobacz **man gcc**. //man grep^

1. **-Wall** - włącza konkretne flagi ostrzegawcze (-Wunused-parameter,-Wno-format-extra-args, ...)
2. **-pedantic –** powoduje wyświetlanie ostrzeżeń przed użyciem alternatywnyc słów kluczowych

- # Zadanie 4

(5pt) Pokaż na przykładzie (np. **sleep 1000**, **sleep 1001**, ...) zarządzanie zadaniami wykorzystując **<polecenie> &** - uruchamianie w tle (background) oraz **jobs, fg, bg, kill** oraz **^Z**. Uruchom kilka zadań w tle i pokaż jak można nimi zarządzać, czyli zatrzymywać, wznawiać oraz kończyć ich działanie. Pokaż jak uruchomione zadanie (nie w tle), można w czasie działania uruchomić w tle np. wykonując komendę **sleep 100** (bez &) w czasie działanie przełącz je do działania w tle.

1. **sleep 1000 & sleep 1001 & sleep 1002**
2. **jobs** – wyświetla zadania
3. **fg** – przesuwa na front
4. **bg –** przesuwa zadanie na tryb w tle
5. **kill** -morduje proces // tutaj warto skorzystać z ps -t i wtedy killem kill -9 pid

- # Zadanie 5

(5pt) Poleceniem **mkfifo** (**man mkfifo**) utwórz potok nazwany (ang. *named FIFO*). Wywołując polecenie **cat** w różnych terminalach spowoduj wpisywanie danych do potoku przez jeden(ne) proces(y), i odczytywanie i wyświetlanie ich przez inne. Zaobserwuj, kiedy polecenie **cat** czytające z potoku czeka na więcej danych, a kiedy kończy pracę. Analogicznie, kiedy czeka na więcej danych (z klawiatury) polecenie **cat** piszące do potoku?

1. Mkfifo x.pipe
2. cat <> x.pipe wyświetla potok na bieżąco
3. multiwpisywanie do jednego naraz działa poprawnie
4. multiwyświetlanie terminale walczą o konkretne sygnały, wyświetlają w sumie całość
5. przyadtne **for i in `seq 20 30` ; do echo $i ; sleep 1; done > namedfifo.pipe**

- # Zadanie 6

- # Zadanie 7

touch ./”nazwapliku”